Transport Canada
Transports Canada

Development
Developpement

TP 7576E

An Expert System for Conflict Avoidance Algorithm Development

Prototype Design

Prepared for

Transportation
Development Centre

Transport Canada

by

CompEngServ Ltd.

MARCH 1986

CA1
DTC 801
86
E87

TP 7576E

AN EXPERT SYSTEM FOR CONFLICT AVOIDANCE ALGORITHM DEVELOPMENT

PROTOTYPE DESIGN

by

John D. Beaton

CompEngServ Ltd.

March 1986

Library and Information Centre / Centre de Documentation (CSLI)
RECEIVED - REÇU LE
AUG 18 1986
Transport Canada / Transports Canada

DISCLAIMER:

The contents of this report reflect the views of CompEngServ Ltd. and not necessarily the official view or opinions of the Transportation Development Centre of Transport Canada.

LANGUAGE NOTICE:

Un sommaire en francais de ce rapport est inclus avant la Table des matieres.

Transport Canada / Transports Canada
Development / Développement

REPORT DOCUMENTATION FORM

1. Transport Canada Report No.	2. TDC Project No.	3. Recipient's Catalogue No.
7576E	6546	

4. Title and Subtitle	5. Report Date
An Expert System for Conflict Avoidance Algorithm Development: Prototype Design	March 1986
	6. Performing Organization Report No. TP85-105
7. Author(s)	8. Transport Canada File No.
J.D. Beaton, B.A. Bowen	D1455-171
9. Performing Organization Name and Address	10. DSS File No.
CompEngServ Ltd Suite 600, 265 Carling Ave. Ottawa, Ontario. K1S 2E1	14SD.T8200-5-5544
	11. DSS or Transport Canada Contract No. 1SD85-00140
12. Sponsoring Agency Name and Address Transportation Development Centre (TDC) 200 Dorchester Blvd West Suite 601, West Tower Montreal, Quebec. H2Z 1X4	13. Type of Report and Period Covered
	14. Sponsoring Agency Code
15. Supplementary Notes	16. TDC Project Officer B. Myers

17. Abstract

The feasibility of applying expert systems technology to the problem of potential flight path conflict situations in a non-radar air traffic control environment was determined. A prototype expert system was designed and built to represent the choice of actions and the decision process as a set of interconnected rules representing the logical functions involved in arriving at a selected option. The system demonstrated that the conflict problem can be solved by emulating the decision-making process of experienced air traffic controllers and that a high level of expertise can be incorporated in such expert systems.

18. Key Words	19. Distribution Statement
Air traffic control, aviation, Conflict avoidance, Expert systems	Limited number of copies available from TDC

20. Security Classification (of this report)	21. Security Classification (of this page)	22. No. of Pages	23. Price
Unclassified	Unclassified		

TDC CDT 79-005
Rev 86

Canada

Transports Canada / Transport Canada

Développement / Development

FORMULE DOCUMENTAIRE DU RAPPORT

1. N° du rapport Transports Canada	2. N° de l'étude — CDT	3. N° de catalogue du destinataire
7576E	6546	

4. Titre et sous-titre	5. Date du rapport
An Expert System for Conflict Avoidance Algorithm Development: Prototype Design	Mars 1986
	6. N° du rapport de l'organisme exécutant: TP85-105
7. Auteur(s): J.D. Beaton, B.A. Bowen	8. N° de dossier Transports Canada: D1455-171
9. Nom et adresse de l'organisme exécutant: CompEngServ Ltd, Suite 600, 265 Carling Ave., Ottawa, Ontario. K1S 2E1	10. N° de dossier — ASC: 14SD.T8200-5-5544
	11. N° de contrat — ASC ou Transports Canada: 1SD85-00140
12. Nom et adresse de l'organisme parrain: Centre de développement des transports (CDT), 200 ouest, boul. Dorchester, Suite 601, Tour ouest, Montréal, Québec. H2Z 1X4	13. Genre de rapport et période visée
	14. Code de l'organisme parrain
15. Remarques additionnelles	16. Agent de projet — CDT: B. Myers

17. Résumé

L'objet de la présente recherche était de montrer la faisabilité d'appliquer les systèmes experts à la solution des problèmes reliés aux conflits de trajectoires dans les espaces aériens qui ne sont pas contrôlés par radar. Le rapport décrit le système expert réalisé en prototype et qui permet de reproduire le processus intellectuel impliqué dans la prise de décisions écrit sous la forme d'une série de règles interreliées représentant autant de fonctions logiques et devant déboucher sur une solution pami toutes les solutions possibles. La recherche a montré que la solution des conflits de trajectoires passe par une reproduction de la démarche intellectuelle des contrôleurs aériens et que les systèmes experts proposés sont capables de performances encore plus étendues.

18. Mots-clés	19. Diffusion
Contrôle de la circulation aérienne, aviation, évitement de conflits, systèmes experts.	Un nombre limité d'exemplaires est disponible du CDT

20. Classification de sécurité (de ce rapport)	21. Classification de sécurité (de cette page)	22. Nombre de pages	23. Prix
Non-classifiée	Non-classifiée		

Canada

SUMMARY

The problem addressed by this contract was that of conflict avoidance in Air Traffic Control (ATC), which involves designing a set of maneuvers that aircraft may execute so that they do not violate predefined separation criteria. This is a problem that requires experts for its solution, because in addition to the many standard rules and regulations (e.g. ATC Manual of Operations) in use, controllers make heavy use of experience and private rules of thumb. Unlike many problems in ATC, the procedures for resolving conflicts are not explicitly stated, as other procedures are. For these reasons, previous solutions based on classical software have failed. However, for these same reasons, an expert system approach is very appropriate, and is capable of solving problems that have been found intractable using classical software.

The purpose of the prototype is to show that expert system technology can be applied to conflict avoidance in sparse airspaces (i.e. airspaces in which a conflict may be avoided without causing another conflict). The prototype shows how the overall problem may be decomposed into a set of logical functions, and how each function can be implemented as an expert system.

The design of the prototype expert system for conflict avoidance is contained in this document, and consists of three parts. The requirements definition gives a detailed description of the requirements and constraints that must be met by the system. These include analyses of the problem feasibility, the system's attributes, its performance requirements, maintenance and growth issues, and acceptance tests. Once these issues have been settled, the system's logical architecture is presented. This defines a global inference algorithm that is used by experts to resolve conflicts, and consists of five functions, each of which represents a subgoal of the overall problem. Since one of these functions consists entirely of classical software, it is not addressed here, although its functional requirements are discussed. The other four functions are implemented as expert systems, and details of each function's implementation are given. The User Interface was implemented using classical software and provides the system's I/O and control functions. It is also described in detail, with a listing given. A set of validation tests are presented which illustrate the system's operation and results. Finally, the extension of this design to a system capable of handling dense airspaces is described.

The results obtained from this design work were primarily required to prove the feasibility of an expert system for conflict avoidance, and were very encouraging. A logical architecture was developed that emulates the decision-making process of an expert in conflict avoidance. Each function in this logical architecture represents a subgoal in the overall decision, and was implemented as an expert system. These functions were connected to form the global inferencing algorithm, which was able to resolve conflicts in a manner analogous to that used by air traffic controllers.

These results prove that expert systems are applicable to the conflict avoidance problem, because an expert system can be built that emulates an expert's decision-making process, and is capable of expansion to very high levels of expertise.

It is felt that the most useful extension of this work would be towards a complete strategic and tactical system for conflict avoidance in dense airspaces. While the extension to dense airspaces would not involve a simple expansion of the current system, a number of concepts have been developed that would allow the solution of the conflict avoidance problem in arbitrarily dense airspaces.

SOMMAIRE

L'objet de la recherche dont il est rendu compte ici est le contrôle de la circulation aérienne, dans l'optique des trajectoires incompatibles pour l'évitement desquelles on voudrait élaborer une série de manoeuvres qu'un aéronef aurait à exécuter, sans compromettre les distances de séparation définies. C'est là une question du ressort des spécialistes en la matière, étant donné les nombreux paramètres en jeu. En effet, outre la règlementation aérienne en vigueur qui est très compliquée, il faut tenir compte aussi des connaissances empiriques que ces spécialistes ont acquises par l'expérience. Contrairement aux autres aspects du contrôle de la circulation aérienne, celui de l'évitement des conflits de trajectoires ne fait pas l'objet de dispositions précises. Ce qui explique pourquoi toutes les solutions proposées jusqu'ici, fondées sur les circuits logiques classiques, n'ont pas donné le résultat escompté. Par contre, et pour ces mêmes raisons, il serait plus avantageux de faire appel aux systèmes experts.

L'objet du prototype réalisé a été de montrer l'utilité d'un système expert dans le but de prévenir les conflits de trajectoires dans les espaces aériens peu encombrés, c'est-à-dire ceux où il est possible de prévenir un conflit de trajectoires sans en provoquer un autre. Le prototype proposé montre comment il est possible de disséquer un problème donné en ses éléments constitutifs qui deviennent autant de fonctions logiques; et comment chacune de ces fonctions peut être écrite sous forme de système expert.

Le présent document sur l'évitement des conflits de trajectoires est en trois parties. La première concerne la définition des besoins, c'est-à-dire la description des exigences et des contraintes devant entrer en ligne de compte, à savoir la faisabilité des systèmes, leurs caractéristiques, les performances attendues, l'entretien, l'objectif de croissance et les essais d'acceptation. Vient ensuite l'architecture logique du système, c'est-à-dire l'algorithme d'inférence destiné à la résolution des conflits et dont les cinq fonctions correspondent à autant de sous-objectifs. Or, étant donné que l'une de ces cinq fonctions est abordée sous forme de circuits logiques classiques, elle ne sera traitée ici que sous l'angle de la mise en oeuvre. Quant aux autres fonctions, elles sont destinées à être écrites sous forme de systèmes experts, de la manière proposée. La fonction Interface usagers-systèmes, abordée sous forme de circuits logiques classiques, sert au traitement et au contrôle entrée-sortie. Sa description détaillée figure ici, accompagnée d'un listage des manoeuvres possibles. Une batterie d'essais de validation est proposée, montrant comment fonctionne le système et mettant en évidence les résultats qu'il est possible d'en tirer. Enfin, le rapport traite de l'extension du système aux espaces aériens encombrés.

Les travaux destinés à démontrer la faisabilité d'un système expert capable de résoudre ce type de conflits ont donné des résultats très encourageants, grâce à une architecture logique qui reproduit la démarche intellectuelle des contrôleurs aériens, et dont les fonctions, correspondant à autant de sous-objectifs, ont été écrites sous forme de systèmes experts. Interconnectées, elles forment l'algorithme d'inférence qui résoud les conflits par un type de raisonnement analogue à celui des contrôleurs aériens.

Ces résultats sont probants car, en plus de démontrer la faisabilité des systèmes experts dans l'application envisagée et dans la reproduction de la démarche intellectuelle des contrôleurs aériens, ils ouvrent la voie à une extension vers des niveaux logiques encore plus élevés.

Nous estimons que l'extension des travaux devrait de préférence s'orienter vers l'élaboration de stratégies et de tactiques exhaustives pour l'évitement des conflits dans les espaces aériens encombrés. Il est vrai que le passage d'un espace aérien peu encombré à un espace encombré ne peut se faire directement. Néanmoins, des préceptes opératoires ont été dégagés permettant l'application du système, sinon aux espaces aériens encombrés, du moins aux espaces aériens arbitrairement assimilés à ceux-ci.

TABLE OF CONTENTS

LIST OF FIGURES xi

LIST OF TABLES xi

1. INTRODUCTION 1

2. APPLICABLE DOCUMENTS 2
   2.1 Precedences of Documents 2
   2.2 List of Documents 2

3. REQUIREMENTS DEFINITION 3
   3.0 Introduction 3
   3.1 Feasibility 3
      3.1.1 Problem Statement 3
      3.1.2 The Proposed System 3
      3.1.3 Problem Suitability 4
      3.1.4 Domain Expert 4
   3.2 Attributes 5
      3.2.1 Physical Attributes 6
      3.2.2 Logical Attributes 6
   3.3 Performance 6
      3.3.1 Domain Performance 6
      3.3.2 User Interface 7
   3.4 Maintenance and Growth 7
   3.5 Acceptance Tests 7
   3.6 Summary 7

4. LOGICAL ARCHITECTURE 8
   4.0 Introduction 8
   4.1 System Level Issues 8
   4.2 The Prototype Architecture 12
      4.2.0 Introduction 12
      4.2.1 Global Inferencing Algorithm 13
      4.2.2 Conflict Classification 13
      4.2.3 Tactic Selection 15
      4.2.4 Tactic Evaluation 17
      4.2.5 Tactic Prioritization 18
      4.2.6 Resolution Procedure Development 18
   4.3 Summary 19

5. CONFLICT CLASSIFICATION 21
   5.0 Introduction 21
   5.1 Conflict Types 21
   5.2 Classification Data Inputs 24
   5.3 Classification Factors 25
   5.4 Classification Rule Base 36
   5.5 Summary 36

6. TACTIC SELECTION 27
6.0 Introduction 27
6.1 Assumptions 27
6.2 Tactic Definitions 28
6.3 Tactic Look-Up Tables.....35
6.4 Tactic Selection Data Inputs 37
6.5 Tactic Selection Factors 37
6.6 Tactic Selection Rule Bases 38
6.7 Summary 39

7. TACTIC EVALUATION 40
7.0 INTRODUCTION 40
7.1 TACTIC EVALUATION DATA INPUTS 40
7.2 TACTIC EVALUATION FACTORS 41
7.3 TACTIC EVALUATION RULE BASES 42
7.4 SUMMARY 42

8. TACTIC PRIORITIZATION 43
8.0 Introduction 43
8.1 Assumptions 43
8.2 Tactic Prioritization Data Inputs 43
8.3 Tactic Prioritization Factors 43
8.4 Tactic Prioritization Rule Bases 44
8.5 Summary 45

9. SYSTEM SHELL 46

10. VALIDATION TESTS 51

11. EXTENSION TO DENSE AIRSPACES 58

APPENDIX A - KNOWLEDGE MODULE RULES

APPENDIX B - FORTRAN SHELL LISTING

APPENDIX C - IMPLEMENTATION CONSIDERATIONS

List Of Figures

Table	Name	Page
1	Conflict Avoidance "Black Box"	9
2	Three Stages of Conflict Resolution	11
3	Prototype System Logical Architecture	14
4	Conflict Types	22
5	Resolution Tactics 1	29
6	Resolution Tactics 2	30
7	Tactic Partitioning	32
8	Tactic Look-Up Table	36
9	Shell Structure	47
10	Shell Operation 1	48
11	Shell Operation 2	49
12	Unconstrained Overtaking Conflict Test	52
13	Constrained Overtaking Conflict Test	53
14	Unconstrained Head-On Conflict Test	54
15	Constrained Head-On Conflict Test	55
16	Unconstrained Crossing Conflict Test	56
17	Constrained Crossing Conflict Test	57

List Of Tables

Table	Name	Page
1	Conflict Classification Data Input	24
2	Implemented Tactic-Conflict Pairs	34
3	Tactic Selection Data Inputs	37
4	Aircraft Characteristics	38
5	Tactic Selection Rule Base Files	38
6	Tactic Evaluation Data Inputs	41
7	Tactic Evaluation Rule Base Files	42
8	Tactic Prioritization Rule Base Files	44
9	Tactic/Factor Weights	45

This page intentionally left blank.

1. INTRODUCTION

This document describes a prototype expert system which was meant to prove the feasibility of expert system technology in the Conflict Avoidance problem domain. Its three main sections present the requirements definition, the system's logical architecture, and its physical implementation. The scope of this prototype included those concepts and capabilities necessary to prove feasibility, and did not attempt to address the full range of knowledge and capabilities required in Conflict Avoidance.

The prototype addresses conflicts involving two aircraft. However, the ideas that are developed for these cases are directly extendable to more complex conflicts (involving three or more aircraft). Also, only expert system technology is addressed by this system, so no classical software has been included beyond what is required to support the expert system. Therefore, the prototype is able to determine how to resolve the conflict, but it does not calculate maneuver details. Adding procedural attachments would help alleviate this omission.

Due to its restricted development time, this system does not include the automatic acquisition of data from present air traffic computer systems, although the system may be upgraded to include this.

2. APPLICABLE DOCUMENTS

The latest issue of the documents listed in section 2.2 prior to the latest issue of this document, forms a part of this design document.

2.1 Precedence of Documents

In the event of conflict between this document and the documents referenced herein, the order of resolution of this conflict shall be :

- This document;
- The documents referenced herein.

2.2 Document List

The following documents are referenced directly or indirectly by this document:

Air Traffic Control Manual of Operations (MANOPS)

ATC Manuals of Training:
- Vol 2 Introduction to ATC
- Vol 3 Flight Progress Strip Marking
- Vol 5 Air Navigation
- Vol 7 Electronic Aids to Navigation

The Practical Application of Criteria for Separation of Aircraft in the NAT Region

Unsolicited Proposal, File No. TP85-105 : "An Expert System for Conflict Avoidance Algorithm Development"

Contract No. 1SD85-00140 : "An Expert System for Conflict Avoidance Algorithm Development"

3. REQUIREMENTS DEFINITION

3.0 Introduction

This chapter presents the requirements definition for the Conflict Avoidance prototype expert system. These requirements were the driving force for the design work that followed, and are presented in five sections. First, the feasibility of implementing a conflict avoidance system using expert system technology is established. Second, the attributes of such a system are defined, since they represent the key requirements and constraints that must be met by the prototype system. It is shown that the conflict avoidance problem does satisfy these attributes. Third, the performance required of the system is defined, since the prototype will not be acceptable until they are met. Fourth, maintenance and growth requirements are stated since they will affect those elements of the system required to support them. Finally, the acceptance criteria the system must meet will be defined.

3.1 Feasibility

The feasibility of applying expert systems to the conflict avoidance problem must be established before work may begin, and this is done on the basis of four points. First, the problem is defined in order to provide a common point of view and subject for analysis. Second, a system that would represent a solution to the problem is proposed. Third, the suitability of the problem is discussed, in that it must satisfy at least one of the requirements of an expert system solution. Finally, the availability of expertise in the form of a recognized expert is established.

3.1.1 Problem Statement

Whenever two or more aircraft are flying in an airspace at the same time, the situation may arise where they will approach each other too closely for safety requirements to be maintained. This situation is called a conflict, and it is the job of Air Traffic Controllers to ensure that conflicts are always prevented. This task is called conflict avoidance, and is a significant problem faced by controllers.

A conflict is defined as the violation of any one of a set of separation criteria between aircraft as defined in the Air Traffic Manual of Operations (MANOPS). The first step in conflict avoidance is detecting the conflict. A conflict must be detected well before it occurs, and information about the conflict must be

available so that a controller can determine how to avoid it. The second step is taking action to ensure that the conflict never occurs. This action consists of developing and implementing a resolution plan or procedure, which consists of a set of maneuvers specifying flightplan modifications for one or more of the aircraft involved in the conflict. This plan is derived by applying some combination of resolution tactics to the particular conflict being considered.

The detection of conflicts has been solved, but past attempts at solving the conflict avoidance problem using classical software have failed. This failure is due to the nature of the problem and the way in which it is solved. Conflict avoidance depends on the expertise of Air Traffic Controllers, and is therefore very difficult (if not impossible) to acquire or specify before a software project begins.

3.1.2 The Proposed System

The acquisition of expertise and it's implementation as software is an area addressed by expert systems technology, and is therefore a suitable approach to the conflict avoidance problem. A prototype system will be built that will prove the feasibility of applying expert systems to this problem by demonstrating correct inferencing over a subset of the problem domain.

The prototype expert system would be used as an advisor to air traffic controllers, and would not act as a replacement for them. While a capable system would be able to handle routine, and even complex, conflict situations (given sufficient system growth), a controller would always be required to validate the system's decisions and to resolve conflicts beyond the system's capabilities. The prototype will assume a sparse airspace (i.e. an airspace in which a conflict can be resolved without inducing another one), such as the Northern Airspace. This allows the conflict avoidance problem to be isolated from other more general concerns a controller must deal with.

This system emphasizes the expert system technology for the selection and development of conflict avoidance tactics. For this reason, it assumes that conflict detection is handled by another system, as this is a solved problem, and that the results of this detection will be available to the prototype. It also does not attempt to solve classical software problems, although they are identified.

3.1.3 Problem Suitability

The conflict avoidance problem is suitable for an expert system implementation because it satisfies the following requirements of an expert system solution.

The conflict avoidance domain is well bounded and well described, as shown by the extensive use of computers in air traffic systems. Air traffic controllers regularly perform the conflict avoidance task, and are therefore qualified to take part in the development of an expert system that would emulate their actions.

The conflict avoidance task is well bounded in terms of the abilities and knowledge required of an air traffic controller, and well defined in terms of what the results of performing this task are. Conflict avoidance must be performed in a fairly short period of time, and does make extensive use of symbolic reasoning in terms of the spatial and temporal relationships between moving aircraft.

The task itself is routinely taught to new controllers, and is decomposed into sub-tasks for this purpose.

Since the proposed system would be used as an advisor to air traffic controllers, as opposed to a replacement for them, it is by definition not threatening to controllers in any way. In fact, it may well alleviate much of the pressure felt by controllers while handling busy airspaces. An expert system would also allow a controller with an adequate level of performance to apply the knowledge and ability of expert air traffic controllers to any conflict situation. Expert solutions presented by the system would also provide on-the-job skill improvement.

Based on these factors, it is obvious that the conflict avoidance problem is suitable for an expert system solution. This assertion will be proved by the design of the prototype system.

3.1.4 Domain Expert

Domain experts who are recognized in their field, familiar with the air traffic domain, the conflict avoidance task, and the user interfaces on air traffic systems, and preferably who have some experience with computer systems must be available for consultation. This computer experience should serve to enhance communication between the domain experts and the design team. It is anticipated that full cooperation will be received from domain experts because of the non-threatening nature of this work.

3.2 Attributes

The prototype's attributes are of two types: physical and logical. Physical attributes are those that affect the expert system builder, language(s), and computers chosen for system implementation. The logical attributes affect the logical structure of the system.

3.2.1 Physical Attributes

This prototype will run on an IBM PC/XT or compatible machine with 640 KBytes of memory, an 8087 math co-processor, and a hard disk with at least 10 MBytes of storage. The system builder used will be TIMM, from the General Research Corporation. The shell will be written using version 1.0 of IBM Professional Fortran by Ryan-McFarland, as required by the TIMM builder.

3.2.2 Logical Attributes

The logical attributes are those characteristics of the system that will affect the three main attributes of expert systems: extendability, simplicity, and explicitness.

The system will be extendable to include more conflict types and tactical solutions. This will require extending the knowledge base, which is conceptually straightforward for an expert system, but it may also require high level language programming to support the expert system.

The knowledge base itself will be very simple since it is available for inspection and modification in the form of rules - a natural representation of many kinds of knowledge. Rule editing facilities within expert system builders provide considerable flexibility within a rule base.

Explicitness of the expert system will be a result of the rule-based knowledge representation.

3.3 Performance

The level of performance that the prototype must exhibit, both in the problem domain and the user interface as described below, must be sufficient to prove that expert system technology is applicable to the conflict avoidance problem.

3.3.1 Domain Performance

This prototype will implement a subset of the complete set of conflict types, tactics, and domain data elements. These must be sufficient to demonstrate the feasibility of applying expert systems to conflict avoidance, and to demonstrate all of the concepts relevant to this application. Note that this system will be given data describing a situation in which a conflict has already been detected. The actual conflict detection can be performed by classical software, and need not be addressed by this system.

3.3.2 User Interface

The user interface will request data from the user as it is required by the inferencing. Data already entered by the user, and the conclusions that could be reached using it, will be shown using a tabular format. Incorrect data should be detected, and an opportunity to correct it given.

3.4 Maintenance and Growth

Since this system is a prototype, it is intended to provide proof of expert system applicability to conflict avoidance, and is not intended to become a large and capable system. However, it must be capable of growth so that additional conflicts and tactics may be implemented. It is recognized that expansion may require some high-level language programming in additional to expanding the rule base.

3.5 Acceptance Criteria/Tests

Test problems that will exercise all of the system's capabilities and each of it's conflicts and tactics will be devised. The solutions given by the expert system will be compared with those given by an expert in conflict avoidance.

3.6 Summary

The requirements for the prototype conflict avoidance expert system have been defined in a format suitable for an expert system implementation. In addition to providing a starting point for the logical design phase, the factors that are particularly relevant to an expert system solution have been exposed. A logical architecture that satisfies these requirements can now be presented.

4. LOGICAL ARCHITECTURE

4.0 Introduction

This chapter discusses the prototype's logical architecture, and is devoted to system-level issues rather than the details of conflict types and resolution tactics. The architecture presented handles single conflicts only. It does not consider future conflicts caused by resolving present ones, and it does not handle more complex conflicts (i.e. those involving more than two aircraft), although it can be extended to cover these cases.

The distinction between the following two terms used throughout this document is important:

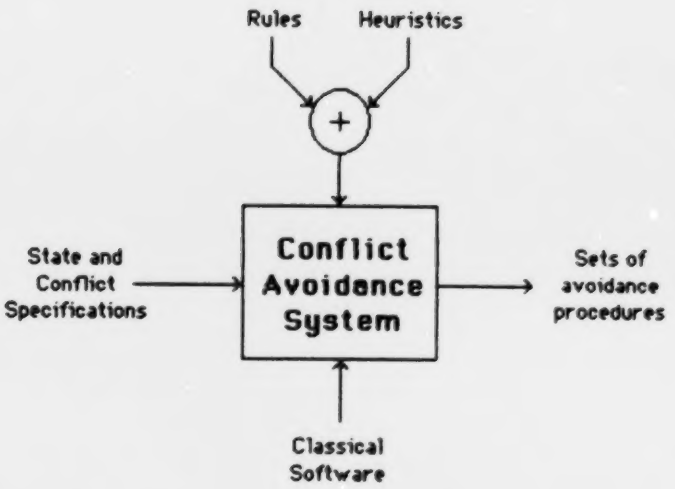
> A flightpath is the space through which an aircraft will fly, and can be thought of as a spatial specification.
>
> A flightplan is a complete spatial and temporal specification of where the aircraft will be at all times, and is a superset of the flightpath.

The temporal aspect of aircraft's flightpaths will have a significant impact on the perspective of conflict avoidance, and the solution designed for it.

4.1 SYSTEM LEVEL ISSUES

Before a logical design can be created, the factors that will affect it's design must be exposed. The first step in exposing these issues will view the Conflict Avoidance Expert System at it's highest level of abstraction as a black box, as shown in Figure 1. It accepts some specification of an airspace as it's input, and provides a procedure for avoiding the conflict as it's output. A set of rules and heuristics are required to guide the system's operation.

The input data would at least include the same data available to an air traffic controller: some statement of aircraft flightplans (analogous to the flight strips currently in use), and data about the airspace. This could be extracted from present-day air traffic computer systems fairly easily. The output data must be made available far enough in advance of the conflict to be of some use, so the system's response time must be reasonably short. Note that any air traffic system that monitors the airspace should be made aware of the flightplan changes caused by the conflict resolution procedures. The other elements



CompEngServ Ltd	Revision : 1.0	Figure 1	Conflict Avoidance "Black Box"
	Date : Jan 26,1986	Drn By : J.D. Beaton	

required by the black box are the rules, heuristics, and classical software procedures that must be acquired from expert air traffic controllers and aviation authorities during system design.

At least three functions must be performed by this system in order to resolve conflicts (see Figure 2). First, the conflict must be detected and characterized. Second, a set of tactics that may be used to solve the conflict must be found. Finally, the details of each tactic's implementation must be determined for the particular conflict and airspace state.

The detection of conflicts has been solved using classical software techniques, so conflict detection will not be considered in this system. However, present conflict detection systems do not characterize conflicts in a way that directly supports expert systems, so this problem will be addressed.

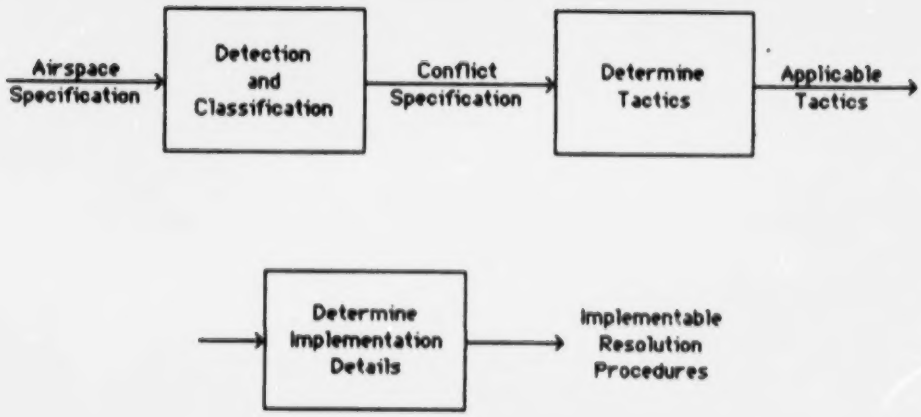
The process of determining resolution tactics may use two different approaches. The system could either select tactics in an incremental fashion, or, given a large number of candidate tactics, it could then eliminate those that are not suitable for the particular conflict to be solved. Both approaches imply a rule set or sets for every tactic that will be considered by the system, and an in-breadth search to consider all possibilities.

The first approach to determining resolution tactics uses a tactic's rule set to determine where a tactic does apply. Therefore, all of the rule sets must be evaluated to determine the applicability of all tactics.

In the second approach, a tactic's rule set determines where it does not apply. A simple look-up table could provide the initial candidates, given the type of conflict to be solved. Then only those tactics chosen from the table, presumably a subset of all of the tactics, would need to have their rule sets evaluated.

While these two methods use opposite approaches, they are related in that if it is known where a tactic does apply, then it is also implicitly known where it does not apply, and vice versa. An operational system should use a hybrid approach to make knowledge acquisition more flexible and efficient. This will also decrease the number of rules, and therefore the system's size and complexity, since if a tactic is applicable in most cases, it is easier to state the few cases where it is not applicable, and vice versa.

Determining a tactic's implementation details is a classical software problem in vector analysis. The specifications (in terms of times, headings, distances, altitudes etc.) for each maneuver in the tactic maut be determined, and a resolution procedure submitted to an air traffic controller. In order for the Determine Tactics function to provide enough information for the implementation details to be unambiguously calculated, it must explore all of the requirements and constraints imposed on the



CompEngServ Ltd	Revision : 1.0	Figure 2	Three Stages of Conflict Resolution
	Date : Jan 23, 1986	Drn By : J.D. Beaton	

tactic by the particular situation it is being applied to.

The implementation of all of these functions depends on whether conflict avoidance will be viewed as introducing transients into aircraft's flightplans (as some literature has assumed), or as reconfiguring the flightplans. Both alternatives must involve local changes to flightplans, and it is possible that this will not involve a change in the future flightplan. However, a flightplan contains temporal as well as spatial specifications, and it is impossible to guarantee that a conflict resolution will not cause an aircraft's schedule to change (i.e. it may well be delayed). Therefore, the temporal specification of a flightplan may have to be permanently changed even by a resolution procedure that only introduces a transient into the flightpath, leading to the conclusion that transient solutions are, in the more general sense, actually flightplan reconfigurations. Since the general view of conflict resolution should have very few or no exceptions, conflict resolution is considered to involve a reconfiguration of one or more of the aircraft's flightplans.

The reconfiguration of flightplans allows more efficient resolutions to be achieved. For instance, an aircraft can be switched to a parallel track and left there until it must be moved again, if indeed it need ever be moved back. A transient solution would require switching it back to it's original track as soon as it's conflict has passed. The first alternative requires fewer maneuvers, and therefore offers higher efficiency.

Constraints could well be added to the possibilities opened up by flightplan reconfiguration once a study of aeronautical operations has been completed. For instance, there may be specific entry and exit points and times for aircraft when passing between control zones, or they may be known and expected ahead of time.

4.2 A Prototype Architecture

4.2.0 Introduction

This section introduces the logical architecture by first describing the algorithm used by the system to resolve conflicts, i.e. it's global inferencing algorithm. Each of the steps, or functions, in this global algorithm is then described in detail.

Each of the following sections is divided in two. The "Implemented" section contains a general discussion of ideas, plus material that describes what this system will implement. The "Proposed" section provides suggestions for upgrading the present system, or building a larger system. Note that the Resolution Procedure Development section has neither division because it will not be implemented.

4.2.1 Global Inferencing Algorithm

The conflict is solved in five steps (see Figure 3), four of which are implemented by this system. Each step answers a question which focusses the system on certain issues.

The first question asked is "What kind of conflict is it?" This ensures that the system will understand the type of conflict it is dealing with, and that it is not hindered by having to take into account irrelevant/extraneous information. Data required to answer this question will include only information about the two aircraft involved.

The second question is "What tactics might be used to resolve this conflict?" These are chosen only on the basis of data about the conflict itself and the aircraft involved. This is analogous to the way an air traffic controller first focusses his attention on the conflict, only widening the scope of his attention when he has determined what might be done. The result of this step is a set of potential tactics, i.e. those tactics that have the potential to resolve the conflict, depending on the state of the airspace.

The third question is "Which of these tactics will actually work?" The scope of the controller's attention has widened to include the airspace and other aircraft in it, so that each potential tactic must result in a safe conflict resolution or be discarded. The result is a set of possible tactics, i.e. tactics with which it is possible to resolve the conflict.

The fourth question is "Which conflict is the best one?" Each of the possible tactics is evaluated to determine which one is "best". This includes efficiency and operational criteria.

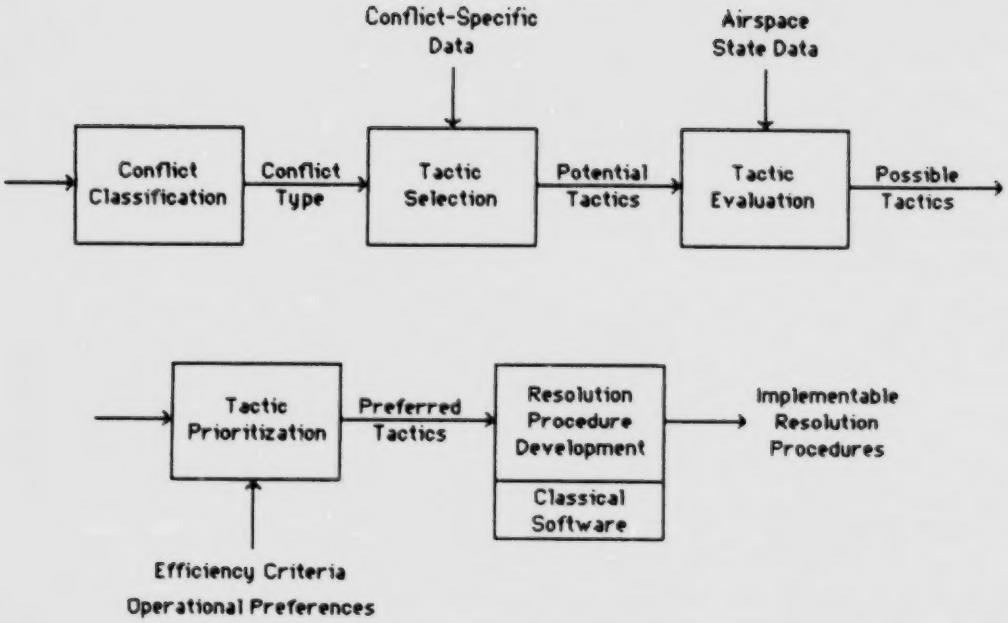
The fifth function would determine the exact maneuver times required to implement the "best" resolution. It would require a considerable amount of classical processing.

4.2.2 Conflict Classification

Implemented

The conflict classification module consists of a single rule base which, when given minimal data about a conflict, determines the type of conflict that will take place.

This function is necessary because classical conflict detection software does not characterize conflicts in the way required by the avoidance system. Therefore, all that a conflict detection routine must provide is basic data about the conflict, which would include the flight level, airspeed, heading, and attitude of both aircraft at the moment of the conflict. Most of the other information required to classify the conflict may be



CompEngServ Ltd	Revision : 1.0	Figure 3	Prototype System Logical Architecture
	Date : Jan 23, 1986	Drn By : J.D. Beaton	14

derived from this data. Note that this may allow the conflict detection software to be simplified, since it need only determine that some conflict will occur, rather than calculating the details of the conflict.

Note that the choice of conflict types depends partly on the tactics that are used to resolve them. In general, a defined conflict type encompasses more than one particular aircraft configuration, where a configuration specifies symbolic relations rather than exact numerical relations. In this sense, a conflict type defines a set of aircraft configurations. It must be assumed that a tactic will be applicable to all configurations within this set (including those at the set's "edges"), regardless of how other factors influence the conflict's resolution.

Proposed

The conflict classification function could be expanded to characterize conflicts in more specific, and symbolic, ways so as to reduce the computational requirements of subsequent functions. It could also pass on it's own input data as required, although this could require some classical software routines to convert data into the particular symbolic form required by the expert system modules.

A larger rule base could use a relatively small rule set that would both determine the type of conflict, and direct the inference engine to another rule set that would characterize the conflict (or do anything other than simple classification) in whatever way might be desireable to provide data for subsequent knowledge modules.

4.2.3 Tactic Selection

Implemented

The purpose of the Tactic Selection function is to determine those tactics that have the potential to solve the specified conflict. This selection process is based only on knowledge about the conflict itself (i.e. all data about the conflict type and the aircraft involved). Data about the airspace itself or other aircraft is not used.

Tactic selection expertise (in the form of rules) may be partitioned according to either conflict types or resolution tactics. This will result in the Tactic Selection rule base being partitioned into a number of rule sets, with one set for either every conflict type or every resolution tactic.

If the rule base is partitioned according to conflict types, (the input) then a rule set may be accessed directly by a conflict type, and only a single rule set need be traversed to

determine applicable tactics. This rule set search must be in-breadth if more than one resolution tactic is to be found.

If the rule base is partitioned according to resolution tactics (the output), then all rule sets must be traversed in order to determine which tactics are applicable to any given conflict type. However, each of these traversals may stop as soon as one rule fires (similar in effect to an in-depth search), if any fire at all.

The traversals of all of these rule sets could be reduced, but not eliminated, by using a simple look-up table to determine which resolution tactics should be explored for a given conflict type. A conflict type could access a row in the table, with each column representing one tactic. Entries would give a yes/no indication as to the applicability of that tactic for that conflict type.

While the conflict type partition might initially appear more computationally efficient, the resolution tactic partition is preferable for this system because it concentrates the knowledge and heuristics for individual tactics in their respective rule sets, making development easier. Note that each of these rule sets could be executed in parallel.

This system uses multiple rule sets that are partitioned according to resolution tactics. Each rule is structured as follows:

```
if   (the conflict type is "x") and
     (some set of conditions is satisfied)
then (tactic "y" is applicable)
```

Note that making the first antecedent the conflict type effectively subdivides each tactical rule set into conflict type subsets. If meta-rules were available to direct the inference engine, only a small part of the tactic's rule set would need to be examined (i.e. only those rules applicable to one conflict type).

Proposed

In a practical system, both partitions should be used, thus partitioning the rule base into tactic-oriented rule sets, and then partitioning each rule set into conflict-type-oriented rule subsets. This could reduce the search time if only the necessary rule bases were examined. Note that all of these rule sets could be compressed into a single monolithic rule base, which would have to be searched in an in-breadth manner so as to expose all of the tactical alternatives.

The calculation of a tactic's implementation details requires specific data regarding times, distances, altitudes, etc. In order for this information to be organized, a frame to

hold this data should be created whenever a tactic is selected. Many of this frame's slots could be filled in based only on conflict data, such as required maneuvers and their execution times. This will be the start of an unambiguous tactic specification, to be completed during Tactic Evaluation (the next function). It will also provide this next function with any specific resolution data it may need. Note that procedural attachments will be required to perform the necessary calculations.

4.2.4 Tactic Evaluation

Implemented

When given a set of tactics that could potentially solve the conflict, the Tactic Evaluation function must eliminate those tactics that can not be used to solve the conflict in the present airspace configuration. The result is a set of possible tactics, i.e. those tactics for which it is possible to determine a complete conflict resolution procedure.

The potential tactics (the input) are consistent and feasible only with respect to the conflict itself. The evaluation of these conflict-specific tactics involves considering all other factors that might affect the implementation of this tactic (i.e. constraints). This would include the location of other aircraft, possible interferences caused by planned maneuvers, airspace restrictions, maximum cruising altitudes, etc. In order to determine some of this data, especially detecting situations like aircraft that will prevent certain maneuvers, classical software routines would be invoked by antecedents within the rule base.

Proposed

The possible tactics resulting from Tactic Evaluation could be an empty set if the airspace imposes enough constraints. However, there must always be a solution for any conflict situation. Therefore, it would be beneficial to consider constraints as being either hard or soft. Hard constraints are those that can not possibly be relaxed, while soft constraints are those that are desirable rather than required. If the Tactic Evaluation function outputs an empty set, some number of the soft constraints could be relaxed to some degree, and the inputs could be reprocessed. This relaxation could continue until a non-empty set is output.

This function would complete the modification of each tactic's frame, again based only on non-conflict data. As an input, the tactic frame would provide the specific information necessary to evaluate the tactic's feasibility with respect to both other aircraft and the airspace in general.

4.2.5 Tactic Prioritization

Implemented

This function will accept a set of possible tactics and essentially sort them in order of preference by assigning a priority to each one. This priority is based on efficiency criteria and operational preferences.

Proposed

Note that some of this information could have been included in the Tactic LUT by including an initial priority for each entry, thus immediately defining preferred priorities. This would initially order the tactics, and present a number of options for the control of subsequent processing. Alternatives would include processing the most desirable tactics first, or processing only the highest priority tactics, with lower priority tactics being processed only if the others turn out to be inapplicable.

Since this function only orders tactics, it would not change any slots in the tactic frames, although it would need to access them. This would allow the Resolution Procedure Development function to be executed in parallel, improving the response time.

4.2.6 Resolution Procedure Development

This function is a problem in classical software, since it can be completely specified as a set of procedures, requiring no heuristic knowledge. Therefore, it is outside of the scope of this system, and is not implemented. However, certain functional requirements that this function should meet have become clear, and will be described here.

This resolution procedure development function is responsible for calculating the details of all maneuvers (i.e. the maneuver specifications) required to resolve the conflict. In order to do this, preceding functions must provide this function with all of the information required to calculate the maneuver specifications. This required data could be different for each tactic, and would depend on the number of maneuvers in the tactic, the characteristics of the conflict situation, etc.

It would be appropriate to create, as soon as a tactic is selected, a data structure for each tactic which will hold all data required by the Resolution Procedure Development function when it processes that tactic. These data structures will be accessible to all preceding functions so that they can add or modify data elements, or use the data elements in their computations. In this system, the Tactic Selection function would insert (into the data structure) initial tactic requirements and constraints in terms of times and distances required for proper aircraft separation, based on the conflict itself. The Tactic

Evaluation function would complete these insertions based on domain data. The Tactic Prioritization function would not modify these data structures, since it only sorts tactics. A sample data structure is shown below:

```
Tactic Name: Change Track - Port
Aircraft   : A

Turn #1
     New Heading    : 340
     Earliest Start: 12:00:00
     Latest Start   : 12:10:00

Distance on new course :       180 nm
Assumed Airspeed       :       720 knots
Time on new course     : 00:15:00

Turn #2
     New Heading    : 000
     Earliest Start: 12:15:00
     Latest Start   : 12:25:00
```

This frame is for a Change Track to Port tactic, and requires two turns. The aircraft was travelling north, so it turned 20 degrees to port for 15 minutes, then back to it's original heading.

It is assumed that the conflict and the airspace will impose time constraints on the maneuvers, forcing them to occur within certain time windows. Each slot in the frame could have a default value, which would be based on tactical requirements and air traffic operations (e.g. assume the earliest possible time for the first maneuver to begin). In order that specifications will be unambiguous, it may be necessary to use distance as well as time, depending on how factors like wind conditions and navigational capabilities affect the flight of aircraft. Finally, a tactic's data structure could have more data elements than are necessary to specify the maneuver. This would allow different aspects of the implementation to be specified in different ways.

It is assumed that this function's processing would be done entirely with classical software, since the resolution procedure is essentially a set of vectors specified in terms of time and space. For this reason, this function is not addressed in this system. It is considered to be in the same class as, although less complex than, conflict detection.

4.3 Summary

This chapter has presented the logical architecture that was created for the prototype conflict avoidance expert system. It defines the global inferencing algorithm as a set of five functions, which guide the system from a problem statement to a

complete solution. The last function is a problem in classical software, so it will not be further addressed. The first four functions are implemented using an expert systems approach, and their physical implementation will be described in the next four chapters.

5. CONFLICT CLASSIFICATION

5.0 Introduction

This chapter defines the implementation of the first logical function; conflict classification. First, the conflict types that will be recognized by the prototype system are defined, and the issues involved in choosing and classifying them are discussed. Subsequent sections describe the data inputs required by this function, the factors used by the rule base, and how the factors are derived from the input data. Finally, the conflict classification rule base, in which the knowledge about the conflict types is stored, is described.

5.1 Conflict Types

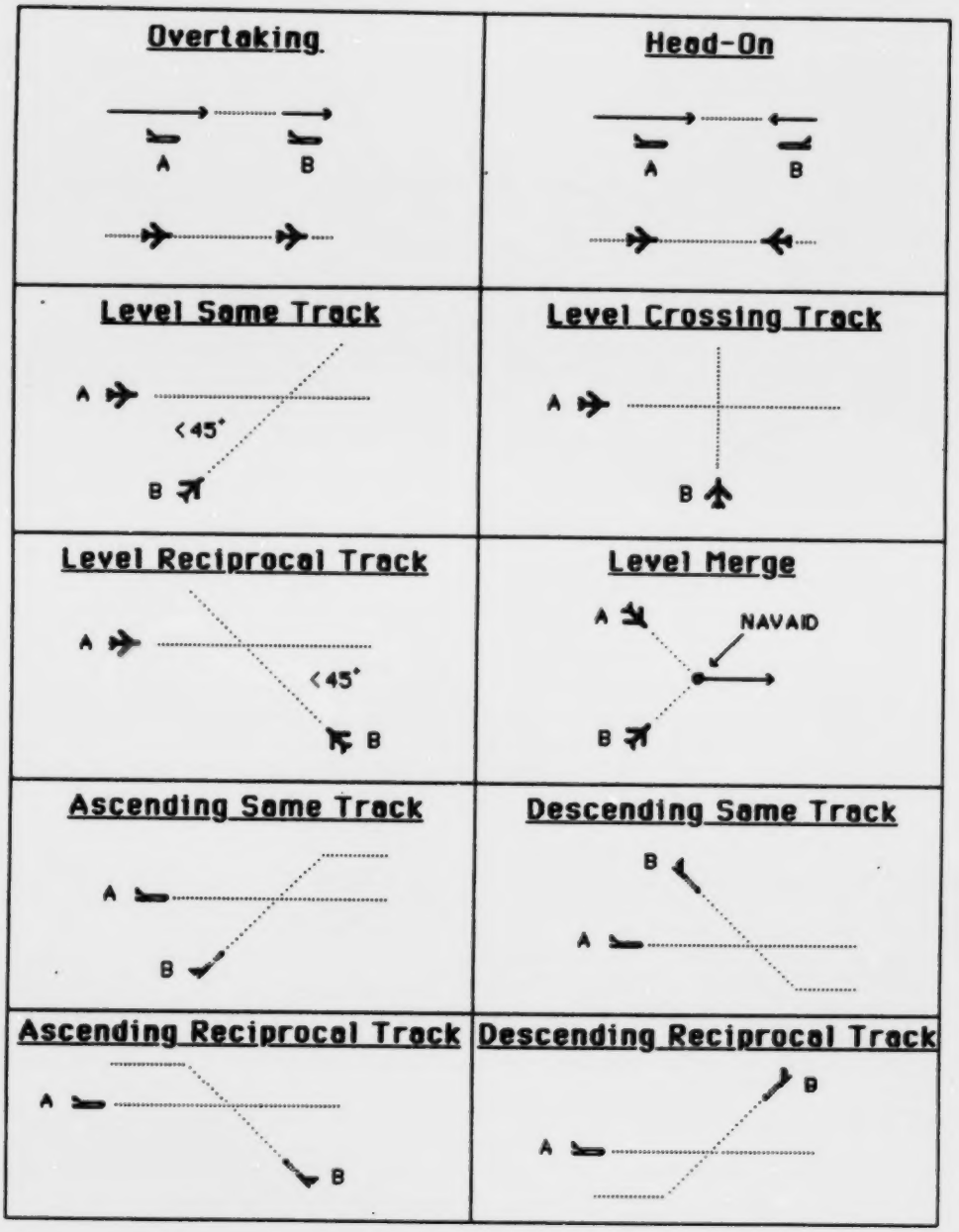
This conflict avoidance system recognizes the 10 conflicts that are shown in Figure 4. They represent the more common conflict situations, and also allow a wide range of resolution tactics to be applied to them. Since these situations have come from the conflict predictor, they do in fact represent conflicts, and the factors used here are sufficient to classify the conflict. They would not be if the situation was not known to represent a conflict (for instance, the overtaking conflict definition would be valid for aircraft on different tracks). Note that situations not recognized by the system are classified as invalid.

In defining the conflict types, only the configuration or state of both aircraft at the moment of the conflict was considered. Maneuvers performed by the aircraft before and after the conflict do not affect the conflict's definition. However, these pre- and post-conflict maneuvers should ideally be considered when evaluating resolution tactics, since they will be used to recognize ways of applying a tactic or improving it's application, thereby increasing the tactic's efficiency. Note that one aircraft has been arbitrarily designated as aircraft "A", and the other as aircraft "B". This provides a reference standard to which rules can be applied in a uniform manner.

The following are brief descriptions of the recognized conflict types, also shown in Figure 4:

1) Overtaking

The aircraft are on the same track, have the same heading, and the following aircraft is faster than the leading aircraft. This includes cases where one aircraft is not exactly behind the other, but slightly to one side or slightly above or below.



CompEngServ Ltd	Revision : 1.0	Figure 4	Conflict Types
	Date : Jan 23, 1986	Drn By : J.D. Beaton	

2) Head-on

The aircraft are on opposite tracks (i.e. the same track and with opposite headings). This includes cases where one aircraft is not exactly in front of the other, but slightly to one side or slightly above or below.

3) Level same track

The aircraft are on the same track, but not identical tracks, and have the same flight level.

4) Level crossing track

The aircraft are on crossing tracks and have the same flight level.

5) Level reciprocal track

The aircraft are on reciprocal tracks, but not opposite tracks, and have the same flight level.

6) Descending same track

The aircraft are on the same track, with one aircraft descending through the other aircraft's track. It is assumed that the descending/ascending aircraft was in level flight before the ascent/descent began, and will be in level flight after it is over.

7) Descending reciprocal track

The aircraft are on reciprocal tracks, with one aircraft descending through the other aircraft's track. It is assumed that the descending/ascending aircraft was in level flight before the ascent/descent began, and will be in level flight after it is over.

8) Ascending same track

The aircraft are on the same track, with one aircraft ascending through the other aircraft's track. It is assumed that the descending/ascending aircraft was in level flight before the ascent/descent began, and will be in level flight after it is over.

9) Ascending reciprocal track

The aircraft are on reciprocal tracks, with one aircraft ascending through the other aircraft's track. It is assumed that the descending/ascending aircraft was in level flight before the ascent/descent began, and will be in level flight after it is over.

10) Level merge

The aircraft are on the same flight level with their tracks initially crossing. They are flying towards a navigational aid, where they will change course.

5.2 Classification Data Inputs

The user is prompted to enter the 10 data elements shown in Table 1. While the aircraft identifications are not strictly necessary here, the start of the conflict avoidance process is an appropriate time to enter them. The airspeed and altitude data will be used to calculate the realtive airspeeds and altitudes used by the rule base.

Table 1 - Conflict Classification Data Input

Variable Name	Purpose	
1) acid_a	Aircraft A's identification	(init. data only)
2) speed_a	Aircraft A's airspeed	
3) alt_a	Aircraft A's altitude	
4) att_a	Aircraft A's attitude	
5) acid_b	Aircraft B's identification	(init. data only)
6) speed_b	Aircraft B's airspeed	
7) alt_b	Aircraft B's altitude	
8) att_b	Aircraft B's attitude	
9) pre_c_angl	Angle between flightpaths before the conflict	
10) post_c_angl	Angle between flightpaths after the conflict (different only if either aircraft turns)	

The post-conflict angle (the angle between the aircraft's flightpaths after the conflict), will differ from the pre-conflict angle if one of the aircraft is turning during the conflict. This situation arises when the aircraft converge on a NAVAID, a not uncommon occurrence, and must be detected.

Two of the factors used in the rule base refer to the attitude of the conflicting aircraft. An aircraft's attitude is it's spatial orientation, which is directly linked to the maneuvers that an aircraft is capable of performing. The attitudes in this system are:

Level	Turning/Banking to Port
Ascending	Turning/Banking to Starboard
Descending	

It is recognized that an aircraft's attitude could be made up of certain amalgams of some of these attitudes, as in bank and

descend. However, for the purpose of this system, increasing the number of attitudes to include much rarer ones would needlessly complicate matters.

5.3 Classification Factors

The set of factors defined below will be used to determine the conflict type, as they represent sufficient information for differentiating between the conflicts used in this system. Note that some classical processing is required for factor preparation, as in determining the relative flight level. These factors could define a large number of conflicts, so that adding more conflicts would probably require few if any added factors.

1) relative flight level	1) A>B (A is higher than B) 2) A=B (same flight level) 3) A<B (A is lower than B)
2) pre-conflict relative heading	1) identical 2) same 3) crossing 4) reciprocal 5) opposite
3) post-conflict relative heading	1) identical 2) same 3) crossing 4) reciprocal 5) opposite
4) aircraft A attitude	1) level 2) descending 3) ascending
5) aircraft B attitude	1) level 2) descending 3) ascending
6) relative airspeed	1) A>B (A is faster than B) 2) A=B (same flight level) 3) A<B (A is slower than B)

Factor Calculation

The processing required to calculate the factor values from the input data is defined here.

Relative Flight Level:

```
A>B  if  (alt_a)-(alt_b)          > min alt diff
A=B  if  abs( (alt_a)-(alt_b) )   < min alt diff
A<B  if  (alt_a)-(alt_b)          < min alt diff
```

Relative Airspeed:

```
A>B  if  (speed_a) > (speed_b)
A=B  if  (speed_a) = (speed_b)
A<B  if  (speed_a) < (speed_b)
```

Relative Heading:

Classify pre_c_angl and post_c_angl using the following categories:

0	identical
1 - 45	same
46 - 135	crossing
136 - 179	reciprocal
180	opposite

5.4 Classification Rule Base

Once the factors for a given conflict situation have been calculated, the expert system is invoked, which will traverse the classification rule base and determine the conflict type. Each conflict type is defined as a set of unique factor values, so there is one rule defining each conflict type recognized by the system. Invalid conflicts are also defined, preventing further processing of aircraft configurations that the tactics were not meant to be applied to.

Conflict classification uses a single knowledge module whose file name is "CLASS", which is fully defined in Appendix B. This knowledge base may be extended to include more conflict types, more factors (to allow different conflict definitions), or more rules.

While the factors in this knowledge base are heavily utilized, some of the conflicts do not require all of the factors for their definition. For example, the post-conflict relative track class is needed only for the "Level Merge" conflict, which is similar to the "Level Crossing Track" conflict. This illustrates that more detailed knowledge is required for finer classifications, but that knowledge can be used only where appropriate - it need not clutter the entire rule base.

5.5 Summary

This chapter has defined the types of conflicts recognized by this system, shown how conflict situations may be classified according to this definition of their types, and also shown how the application of resolution tactics depends on this classification.

6. TACTIC SELECTION

6.0 Introduction

This chapter defines the system's method of selecting tactics for a given conflict type, which involves using an expert system to choose a workable set of tactics from an initial long list of tactics based only on information about the conflict itself and the aircraft involved in it.

Initially, the focus of tactic selection is clarified in Section 6.1 by stating the assumptions that affect the implementation of this function. A suggested set of tactics that the system could implement, and the conflict-tactic pairs that have been implemented, are then discussed. Section 6.3 then shows how an initial long list of tactics is derived. Subsequent sections describe the data inputs required by this function, the factors used by the rule base, and how the factors are derived from the input data. Finally, the tactic selection rule base, in which the knowledge about the tactics is stored, is described.

6.1 Assumptions

All conflict avoidance tactics fall into one of two categories: they must either implement a new maneuver, or modify an existing maneuver. Regardless of the category, the maneuver must be implemented far enough in advance of the conflict to ensure that it is avoided.

This system will address the former situation, where a resolution tactic is determined that does not attempt to integrate any of the aircraft's pre-conflict or post-conflict maneuvers. These pre-conflict and post-conflict maneuvers can affect the choice of a tactic, but they will not affect it's implementation.

Modifying an existing maneuver involves not only choosing a set of tactics that could resolve the conflict, but also trying to merge them with existing maneuvers. However, it is not known how far ahead of the conflict the detection will be, when the pre-conflict maneuver will actually take place, or how much time is required to determine and implement the resolution procedure. This added complexity would obscure the benefits of expert systems technology more than illustrate them, so this approach was not considered here. However, this approach is well within the capabilities of an expert system, and should be included in any future work that is done.

The exception to these assumptions regards the modification

of a maneuver being performed by one of the conflicting aircraft during the conflict. For example, if an aircraft is descending when it is in conflict, the conflict can be avoided by simply delaying the aircraft's descent. This is based on the idea that such a maneuver would be initiated shortly before (i.e. close to the start of) the conflict, and that the conflict would be detected far enough in advance for this maneuver to be modified. This is a valid assumption, since there must always be enough time before the conflict to implement a new resolution tactic (i.e. one that was not originally in the aircraft's flightplan), and this would, in general, require even more lead time than modifying a maneuver that the aircraft would already be executing during the conflict.

The second assumption is that an aircraft maneuvering at the time of the conflict will not be required to execute a different maneuver concurrently with the first, since this system does not recognize multiple simultaneous aircraft attitudes (e.g. bank and descend at the same time). Note that this does not exclude a simple modification of the original maneuver, such as delaying a descent.

6.2 Tactic Definitions

Figures 5 and 6 show the resolution tactics, all of which appear in the Tactic Look-Up Table (LUT). Each of these tactics carries with it certain factors (i.e. assumptions, requirements and constraints regarding both the system and the domain) that will affect it's application to conflicts. It is important that these factors be explicitly stated, because they are partly dependent on changing domain data such as aircraft navigational capabilities. While the following discussion is not exhaustive, it does illustrate the importance of an explicit statement of factors.

As shown below, tactics may be partitioned into two groups: those that implement new maneuvers, and those that modify existing maneuvers.

New Maneuvers:	Maneuvers Modifications:
1) Change Flight Level	7) Delay Altitude Change
2) Change Airspeed	8) Advance Altitude Change
3) Change Track	9) Change Turn Time
4) Delay Vector	10) Cut Corner
5) Vertical Detour	
6) Holding Pattern	

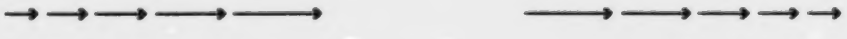
Tactics that implement new maneuvers should be considered in light of maneuvers already in the aircrafts' flightplans, since these existing maneuvers will affect both the feasibility and efficiency of new maneuvers. Tactics that modify existing maneuvers require, as per the second assumptions in Section 6.1, that one of the aircraft be performing that maneuver (i.e. the

Figure 1a Change Flight Level

Climb to New Flight Level

Descend to New Flight Level

Figure 1b Change Airspeed



Increase Airspeed

Decrease Airspeed

Figure 1c Change Track

Change to Port

Change to Starboard

Figure 1d Delay Vector

Delay Vector Port

Delay Vector Starboard

Figure 1e Vertical Detour

Detour Over

Detour Under

Figure 1f Delay Altitude Change

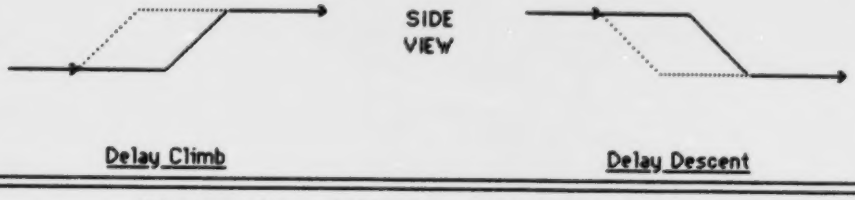


Figure 1g Advance Altitude Change

Figure 1h Change Turn Time

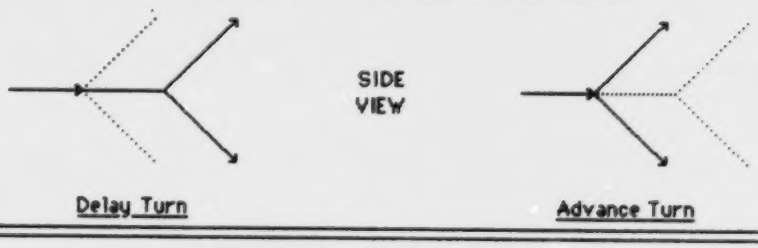


Figure 1i Cut Corner

Figure 1j Holding Pattern

one implicit in the tactic) at the conflict point.

A second partitioning of tactics may be done depending on the effect they have on the flightpath. If the resolution tactic imposes only temporary changes in the aircraft's flightpath (i.e. a flightpath transient), then it ultimately only delays the aircraft. However, if the resolution tactic results in the aircraft flying on a different flightpath (i.e. a flightpath reconfiguration), then a future maneuver may be required or requested (e.g. by the aircrew) in order to restore the aircraft to it's original flightpath. In either case, the tactic itself and/or future maneuvers imposed by the tactic could be optimized by examining future maneuvers in the aircraft's original flightplan to see if they could be merged. Figure 7 shows these two partitions, and how the resolution tactics are classified using them.

Tactics implementing new maneuvers:

(1) Change Flight Level - Up/Down

This tactic will only be applied to an aircraft when it is in level flight. It is not applied to an ascending or descending aircraft, because the aircraft is already changing flight levels. It is not applied to banking aircraft because this would require a simultaneous bank and ascend/descend, which is not recognized by the system. Before changing the flight level, the system must ensure that there are no aircraft above/below the one to be moved, and that the aircraft is capable of climbing or able to descend.

(2) Change Airspeed - Increase/Decrease

This tactic will effectively get an aircraft to the conflict zone earlier or later, but it will not change it's flightpath. Since aircraft usually fly at their desired and/or optimal speeds, changing the airspeed could decrease efficiency by reducing the aircraft's operating efficiency or increasing passenger/pilot dissatisfaction (this assumes that keeping passengers and pilots happy, i.e. getting them to their destination on time, is important). Also, an aircraft might not be capable of flying faster or slower if it is already at it's operating limit.

(3) Change Track - Port/Starboard

This tactic also is only applied to an aircraft when it is in level flight. Two turns are required to switch to an adjacent track, which must be free of other aircraft.

	Flightplan Transient	Flightplan Reconfiguration
Tactics Implementing New Maneuvers	Change Airspeed Delay Vector Vertical Detour Holding Pattern Cut Corner	Change Flight Level Change Track
Tactics Modifying Existing Maneuvers	Delay Altitude Change Advance Altitude Change	Change Turn Time

(4) Delay Vector - Port/Starboard

While it requires four turning maneuvers to implement, and results in a local change only in the flightpath, the net effect of a delay vector is to make the aircraft arrive at the conflict zone later than it would have. This tactic is applied to an aircraft when it is in level flight, and the adjacent track (at the same flight level) must be available for a short time.

Note that this same maneuver could be used, as a lateral detour, to go around something, like the other aircraft in a head-on conflict. When used for this purpose, a different set of criteria would apply to it's final implementation, since the purpose would be to make sure the aircraft was far enough away from it's original flightpath for a certain period of time, rather than ensuring it is delayed on it's original flightpath for a certain period of time.

(5) Vertical Detour - Up/Down

This tactic is similar to the Delay Vector in that it requires four manuevers to implement and results in a local change in the flightpath. While it could be used as a simple delay, it is assumed that changing altitudes for this purpose would be very inefficient. Therefore, it is used like a Lateral Detour would be - to "hop over" or "dip under" another aircraft. Aircraft ascend and descend feasibility and clear detour tracks are important factors in this tactic's implementation.

(6) Holding Pattern - Port/Starboard

While a holding pattern is applicable to any conflict, it is very inefficient for enroute flight, and should be used only when absolutely necessary. It's net result is like that of a Delay Vector.

<u>Tactics that modify existing maneuvers:</u>

(7) Delay Altitude Change - Ascent/Descent

This tactic refers exclusively to an altitude change initiated just before the conflict. It allows the conflicting aircraft to pass under or over, thus avoiding the conflict.

(8) Advance Altitude Change - Ascent/Descent

This tactic refers to an altitude change that would be executed after the conflict, and would involve one aircraft changing to some flight level other than the conflicting aircraft's flight level. This allows the conflicting aircraft to

pass under or over, thus avoiding the conflict.

(9) Change Turn Time - Early/Late

This tactic requires that at least one of the aircraft has a route turn reasonably close to the conflict, either before or after it. Changing the turn time would result in the aircraft flying on a parallel track to the one intended. Note that for this system only an early turn tactic is appropriate because only a turn taking place after the conflict may be modified.

This conflict type usually occurs over a NAVAID, which is a preferred turning point. Changing the time, and therefore the location, of the turn would complicate navigation, and may be undesireable for some aircraft.

(10) Cut Corner

If the conflict occurs soon before or after a course change, then the conflict zone could be avoided entirely by "cutting the corner" of the turn using 3 maneuvers. This is similar to an advanced/early turn, but eventually returns the aircraft to it's flightpath. This would result in local flightpath changes, but only minor delays in the future flightplan.

Implemented Tactics

The tactic-conflict pairs that are examined by the prototype are shown below. Time constraints prevented more conflict-tactic pairs from being implemented, but those being implemented are sufficient for meeting the system's goals. Note that only tactics involving changes in flight level are used for the level crossing track conflict because other tactics are used no more than 10 to 20% of the time in practice. The "change airspeed" tactic will obviously not resolve a head-on conflict, and the "change flight level down" tactic was not implemented for overtake or head-on conflicts because of time limitations.

Table 2 - Implemented Tactic-Conflict Pairs

Tactic		Conflicts		
Change Flight Level Up	- A	Overtake	Head-on	Crossing
Change Flight Level Up	- B	Overtake	Head-on	Crossing
Change Airspeed, Increase	- B	Overtake	-------	--------
Change Track to Port	- A	Overtake	Head-on	--------
Change Track to Port	- B	Overtake	Head-on	--------
Change Flight Level Down	- A	--------	-------	Crossing
Change Flight Level Down	- B	--------	-------	Crossing

6.3 Tactic Look-Up Table (LUT)

The Tactic LUT (see Figure 8) represents the first pass at selecting tactics for conflict resolution, and results in a "long list" based on fundamental conflict and tactic characteristics in which incompatible conflict-tactic pairs are quickly eliminated (e.g. decreasing airspeed in a head-on conflict is pointless).

This approach is advantageous for two reasons. First, while this long list might include almost all of the tactics, the elimination of even one tactic results in a significant speedup, because any one tactic's rule base would require considerable computation to traverse, while table look-ups are very fast. Second, the Tactic LUT also has the advantage of being simple to implement and update.

A table entry in the Tactic LUT could be one of four symbols, as shown below:

X - this tactic is not applicable to this conflict type
A - this tactic is applicable only to aircraft A
B - this tactic is applicable only to aircraft B
o - this tactic is applicable to either aircraft (A or B)

In addition to one of these symbols, a certainty factor giving the tactic's likelihood of success or a relative order of preference could be included. Since there are certain tactics that are preferred for resolving certain conflicts, this would provide an initial and "natural" (in that air traffic controllers do this) ordering or prioritization of tactics.

There are a number of issues regarding the Tactic LUT. First, although the Delay Altitude Change and Change Turn Time Early tactics are included in the table, they are not suitable for any of the conflict types because they would affect only maneuvers occuring before the conflict, and it is assumed that this system will not do this. Second, note that some maneuvers can be used for only one of the aircraft. For instance, in an Overtaking conflict, the Increase Airspeed tactic can only be applied to the leading aircraft. Finally, some tactics are obviously inefficient in their application. Using a Vertical Detour tactic for a Lateral Non-Intersection conflict is feasible, but very inefficient considering that other tactics that do not require altitude changes could be used.

Table Entries

X - not applicable
A - applies to A only
B - applies to B only
o - applies to A or B

Tactics:		Conflicts: Overtaking	Head-On	Level Same Track	Level Crossing Track	Level Reciprocal Track	Descending Same Track	Descending Reciprocal Track	Ascending Same Track	Ascending Reciprocal Track	Level Merge
Change FL	Up	o	o	o	o	o	X	A	A	X	o
	Down	o	o	o	o	o	A	X	X	A	o
Change Airspeed	Increase	B	X	o	o	o	o	o	o	o	o
	Decrease	A	X	o	o	o	o	o	o	o	o
Delay Vector	Port	o	o	A	o	B	A	A	A	A	o
	Starboard	o	o	B	o	A	A	A	A	A	o
Change Track	Port	o	o	B	B	B	A	A	A	A	X
	Starboard	o	o	A	A	A	A	A	A	A	X
Change Turn Time	Early	o	o	o	o	o	A	A	A	A	o
	Late	X	X	X	X	X	X	X	X	X	o
Delay Altitude Change	Ascent	X	X	X	X	X	X	X	B	B	X
	Descent	X	X	X	X	X	B	B	X	X	X
Advance Altitude Change	Ascent	o	o	o	o	o	X	X	B	B	o
	Descent	o	o	o	o	o	B	B	X	X	o
Vertical Detour	Over	o	o	o	o	o	A	A	A	A	X
	Under	o	o	o	o	o	A	X	X	A	X
Holding Pattern	Port	o	o	o	o	o	o	o	o	o	o
	Starboard	o	o	o	o	o	o	o	o	o	o

6.4 Tactic Selection Data Inputs

The system prompts the user for following 4 data elements shown in Table 3. These are directly used as factors in the tactic selection rule bases.

Table 3: Tactic Selection Data Inputs

Variable Name	Purpose
A's Next Maneuver	Aircraft A's first post-conflict maneuver
A's Priority	Aircraft A's urgency or priority
B's Next Maneuver	Aircraft B's first post-conflict maneuver
B's Priority	Aircraft B's urgency or priority

6.5 Tactic Selection Factors

The following factors used by the rule base are calculated from the input data and from aircraft characteristics in Table 4. These characteristics are accessed by the aircraft identification which was input during conflict classification.

A's Next Maneuver
B's Next Maneuver
A's Priority
B's Priority

Is A able to ascend?
Is B able to ascend?
Is A able to descend?
Is B able to descend?
Is B able to increase airspeed?

Is A's current airspeed > B's maximum airspeed ?

A's Navigational Capabilities
B's Navigational Capabilities

Factor Calculation

Is A able to ascend?
yes if current altitude < max altitude

Is B able to ascend?
yes if current altitude < max altitude

Is A able to descend?
yes if A's current altitude > minimum allowable altitude

Is B able to descend?
yes if B's current altitude > minimum allowable altitude

Is B able to increase airspeed?
yes if current airspeed < max airspeed

Aircraft Characteristics

This data is used to calculate factors for tactic selection and prioritization, and is dependent only on the type of aircraft, rather than the situation in which the aircraft finds itself. They have been predefined for the aircraft types used in this system, and are accessed using the aircraft ID.

Table 4 Aircraft Characteristics

Aircraft Type	Boeing 767	C5A Galaxy	Cessna 150	CF-18
Aircraft ID	B767	C5AGal	Cess	CF-18
Optimum airspeed	650	400	155	850
Maximum airspeed	750	500	200	2000
Minimum airspeed	120	125	65	150
Optimum altitude	FL290	FL220	FL100	FL300
Maximum altitude	FL350	FL300	FL150	FL500
Maneuverability:	nominal	low	nominal	nominal
Nav. Capabilities	nominal	nominal	low	nominal

6.6 Tactic Selection Rule Bases

Table 5 lists the knowledge modules used for tactic selection, which are fully defined in Appendix B. They are invoked according to the conflict type. Note that some of the knowledge bases are used when applying a tactic to either A or B. This is possible because the rules applied in either case are the same, but the input data is changed to reflect whichever aircraft is being examined.

Table 5 Tactic Selection Rule Base Files

File Name	Conflict	Tactic	Aircraft
OSCHFLU	Overtake	Change Flight Level Up	A or B
OSCHASIB	Overtake	Increase Airspeed	B
OSCHTRP	Overtake	Change Track to Port	A or B
HSCHFLU	Head-On	Change Flight Level Up	A or B
HSCHTRP	Head-On	Change Track to Port	A or B
CSCHFLU	Crossing	Change Flight Level Up	A or B
CSCHFLD	Crossing	Change Flight Level Down	A or B

6.7 Summary

This chapter has shown that a long list of tactics may be selected, based only on information about the conflict, from knowledge about the tactics and the conflict. This function has been implemented as an expert system, with classical software added in the form of the Tactic LUT. The next function will add airspace data to determine which tactics are capable of resolving the tactic.

7. TACTIC EVALUATION

7.0 Introduction

This chapter describes the system's tactic evaluation function, which evaluates potential tactics with respect to the airspace and other aircraft in order to determine possible tactics that may be used to resolve the conflict. The chapter's sections describe the data inputs required by this function, the factors used by the rule base, and how the factors are derived from the input data. Finally, the conflict classification rule base, in which the knowledge about the conflict types is stored, is described.

7.1 Tactic Evaluation Data Inputs

The data describing the state of the airspace that is used in the tactic evaluation knowledge bases is highly symbolic in nature, as required by the expert-system-only orientation of this system. For instance, the "Track to port occupied" factor is used to determine whether or not one of the aircraft may be switched onto that track, and must be calculated through the use of a procedural attachment (or provided by the user, as in this system). This is similarly true for all of the factors used.

The system prompts the user for the 7 data elements shown in Table 6. The tracks are either occupied or not occupied. The weather conditions are only one of "bad to port of A", "bad to starboard of A", "bad above", or "OK", while a restricted airspace may exist to port of A, to starboard of A, or not at all.

It is assumed that the aircraft are flying on "tracks" solely for conceptual simplicity. In an operational system, a classical software procedure would calculate where the airspace around the conflict was "occupied" by other aircraft, in the sense that moving one of the conflicting aircraft in that direction would cause another conflict. Imposing a track structure on the airspace simplifies the visualization of this concept. Note that determining the status of these "tracks" is very similar to conflict detection, which is a solved problem in classical software.

While in actual fact bad weather conditions or restricted airspaces may exist in numerous places relative to the conflict area, specifying only one location for each is sufficient to show how factors related to the state of the airspace will affect the tactic evaluation process. In an operational system, these factors would have complete flexibility to completely describe the state of the airspace.

The status of the "tracks" will be calculated in a manner similar to that used for conflict detection, so that the same data will be used. It is assumed that this data, along with the weather conditions and locations of restricted airspaces, could be extracted automatically from current air traffic control computer systems.

Table 6: Tactic Evaluation Data Inputs

Variable Name	Purpose
htstat	Track Above Status
ltstat	Track Below Status
tastat	Track Ahead Status
tpstat	Track to Port of A Status
tststat	Track to Starboard of A Status
weather	Weather Conditions
rastat	Restricted Airspace Status

7.2 Tactic Evaluation Factors

The following factors used by the rule base are calculated from the input data shown in Table 5.

Higher Track
Lower Track
Track Ahead

Track to Port of A
Track to Port of B

Weather Conditions
Weather Conditions to Port of A
Weather Conditions to Port of B

Restricted Airspace
Restricted Airspace to Port of A
Restricted Airspace to Port of B

Factor Calculation

Track to Port of B = Track to starboard of A

Weather Conditions to Port of A
= bad if weather conditions = bad to port of A
= OK if weather conditions = any other value

Weather Conditions to Port of B
= bad if weather conditions = bad to starboard of A

= OK if weather conditions = any other value

Restricted Airspace to Port of A
= yes if restricted airspace = bad to port of A
= no if restricted airspace = any other value

Restricted Airspace to Port of B
= yes if restricted airspace = bad to starboard of A
= no if restricted airspace = any other value

7.3 Tactic Evaluation Rule Bases

Table 7 lists the knowledge modules used for tactic selection, which are fully defined in Appendix B. They are invoked according to the conflict type. Note that some of the knowledge bases are used when applying a tactic to either A or B. This is possible because the rules applied in either case are the same, but the input data is changed to reflect whichever aircraft is being examined.

Table 7 Tactic Evaluation Rule Base Files

File Name	Conflict	Tactic	Aircraft
OECHFLU	Overtake	Change Flight Level Up	A or B
OECHASIB	Overtake	Increase Airspeed	B
OECHTRP	Overtake	Change Track to Port	A or B
HECHFLU	Head-On	Change Flight Level Up	A or B
HECHTRP	Head-On	Change Track to Port	A or B
CECHFLU	Crossing	Change Flight Level Up	A or B
CECHFLD	Crossing	Change Flight Level Down	A or B

Each of the tactic evaluation rule bases checks the conflict situation with respect to the tactic being applied to determine whether or not certain criteria are satisfied. The criteria used may be different for each tactic. For instance, the weather conditions to port of A are only relevant when applying the tactic "Change Track to Port" to aircraft A.

7.4 Summary

This chapter has shown that an initial long list of tactics may be evaluated with respect to non-conflict-specific factors in order to determine which tactics will be able to resolve the conflicts. This evaluation is implemented as a set of knowledge modules emulating the expert's ability to perform this same function. All that remains to be done is for the tactics to sorted in order of preference, so that the "best" tactic may be used.

8. TACTIC PRIORITIZATION

8.0 Introduction

This chapter describes the system's method of sorting the tactics in order of preference by assigning a priority to each possible tactic. First, the assumptions that affect this function's implementation are discussed. Subsequent sections describe the data inputs required by this function, the factors used by the rule base, and how the factors are derived from the input data. Finally, the conflict classification rule base, in which the knowledge about the conflict types is stored, is described.

8.1 Assumptions

The method of ordering tactics used here assumes that no further data will be acquired (i.e. only data that has already been acquired will be used). This could be changed in future systems as the factors used by air traffic controllers are determined. Note that while this function could be merged with preceding functions, it is separated here to explicitly show it's utility and functioning.

8.2 Tactic Prioritization Data Inputs

The user is not required to input any data for the tactic prioritization phase. The system only uses data that has already been defined. This absence of data input is not an architectural imperative, but an implementation preference. An expanded system could use previously unused data in this phase.

8.3 Tactic Prioritization Factors

The following factors are used by the prioritization rule bases:

Maneuverability
Navigational Capability
A/B Above Optimum Altitude
B At Optimum Airspeed
A/B Next Maneuver To Turn Right

The maneuverability refers to the pilot's willingness to maneuver based on whether or not it is possible and/or desireable for him to perform the required maneuver. This factor also depends on the aircraft's flight characteristics and

capabilities. Factors referring to optimum altitude and airspeed should, in practice, be based on pilot feedback, since these factors are generally computed either by on-board flight management systems or by the aircrew. While this makes the inclusion of these factors debatable, they still represent considerations that are in fact made when determining which tactic is the best. Note that these factors could be calculated by ground-based systems, or they could be calculated on board the aircraft and automatically transmitted to a ground-based system. Note also that airspeed and groundspeed must be differentiated, as they can cause confusion.

Factor Calculation

The first four factors are calculated using aircraft characteristic data from Table 4. The current groundspeed and altitude are compared with optimal values to calculate factors 3 and 4. The last factor is based on data which was input during the Tactic Selection. While it is recognized that calculating aircraft characteristics is potentially very complex, including these types of factors show that this type of knowledge can be included in an expert system, with complex calculations being handled by procedural attachments.

8.4 Tactic Prioritization Rules

Table 8 lists the knowledge modules used for tactic selection, which are fully defined in Appendix B. They are invoked according to the conflict type. Note that some of the knowledge bases are used when applying a tactic to either A or B. This is possible because the rules applied in either case are the same, but the input data is changed to reflect whichever aircraft is being examined.

Table 8 Tactic Prioritization Rule Base Files

File Name	Conflict	Tactic	Aircraft
OPCHFLU	Overtake	Change Flight Level Up	A or B
OPCHASIB	Overtake	Increase Airspeed	B
OPCHTRP	Overtake	Change Track to Port	A or B
HPCHFLU	Head-On	Change Flight Level Up	A or B
HPCHTRP	Head-On	Change Track to Port	A or B
CPCHFLU	Crossing	Change Flight Level Up	A or B
CPCHFLD	Crossing	Change Flight Level Down	A or B

Each tactic is given a base weight, which reflects it's priority independent of other factors, and includes factors such as the "first come first served" rule. Each tactic has a number of other factors related to it, each of which has a weight associated with it. If a factor does apply to a given conflict situation, the tactic's base weight is multiplied by the factor's

weight. The tactic's final priority is determined by conditionally multiplying it's base weight by the weight of all of the tactics that affect it. The weights of all tactics and all factors is shown in Table 9. These are illustrative values, and are open to examination and modification by expert air traffic controllers.

In the Overtake and Head-On conflicts, the first-come first served "rule" makes aircraft A the preferable target for a maneuver, since it is assumed to arrive at the conflict first. In the Head-On conflict, neither aircraft arrives first, so other factors decide the priority.

Table 9 Tactic and Factor Weights

Tactic/Factor	Weight
Overtake, Change Flight Level Up - A	1.0
Overtake, Change Flight Level Up - B	0.7
Overtake, Increase Groundspeed - B	0.7
Overtake, Change Track to Port - A	1.0
Overtake, Change Track to Port - B	0.7
Head-On, Change Flight Level Up - A	1.0
Head-On, Change Flight Level Up - B	1.0
Head-On, Change Track to Port - A	1.0
Head-On, Change Track to Port - B	1.0
Crossing, Change Flight Level Up - A	1.0
Crossing, Change Flight Level Up - B	0.7
Crossing, Change Flight Level Down - A	1.0
Crossing, Change Flight Level Down - B	0.7
Maneuverability	0.8
Navigational Capability	0.8
A/B Above Optimum Altitude	0.9
B At Optimum Airspeed	0.9
A/B Next Maneuver To Turn Right	0.9

8.5 Summary

This chapter has shown how tactics may be sorted in order of preference based on efficiency criteria and operational parameters. This ordered list of tactics may now be presented to a controller for implementation.

9. SYSTEM SHELL

The system shell provides two major functions: first, it provides a user interface, and second, it co-ordinates all of the TIMM knowledge modules.

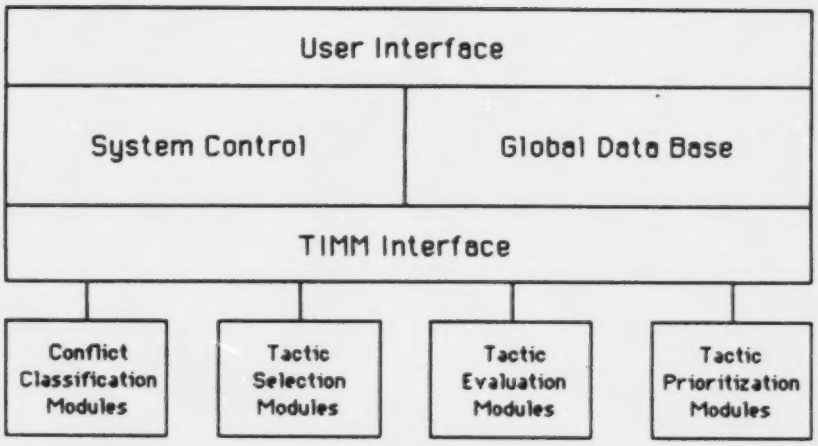
As a user interface, it prompts the user for data input, and provides data output as results are calculated by the system. The data input uses a pseudo menu-driven format, where the input choices are explicitly listed, and the user enters the number corresponding to the desired input datum. If conflicts have been previously resolved in the same "run" of the shell, the data used for the previous run is shown when the user is prompted for input data. If the same data is to be used again, the user merely types "-" and <CR>. This saves data entry effort during multiple runs of similar conflicts. All data output uses the same tabular format. The user is given an opportunity to check and re-enter data before it is used by the system, to minimize the possibility of error.

The shell is also required for co-ordinating all of the knowledge modules. For each phase of the conflict resolution, the knowledge modules use a common set of factors. However, each module may use a different subset of factors, so the shell must pass each module the proper factors in the proper order. The shell must also invoke only those modules that are required, as defined by the conflict type and the potential and possible tactics. Note that the shell also calculates the factors used by the knowledge modules. This shows that the factors used by a conflict avoidance expert system may be quite different in form, but not drastically different in overall content, from the data available to the system.

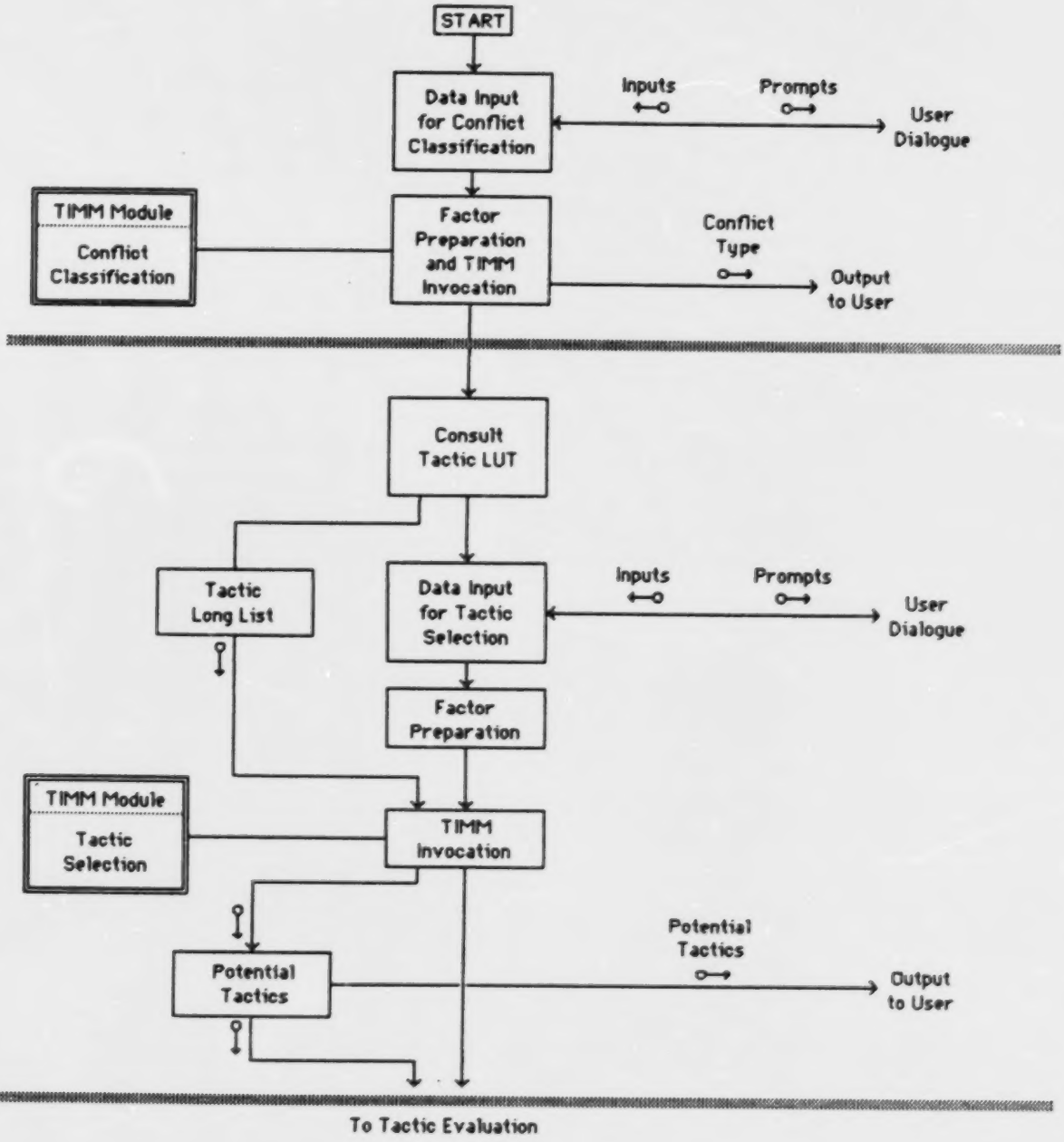
The overall structure of the system shell is shown in Figure 9. The user interface provides data I/O prompting and formatting. The system control function provides overall sequencing for the system. The global data base holds all data required for system operation, and the TIMM interface calls the TIMM modules and passes data to them as it is requested.

Figures 10 and 11 show the four distinct phases in the system shell's operation: Conflict Classification, Tactic Selection, Tactic Evaluation, and Tactic Prioritization. Each phase follows the same general sequence of steps:

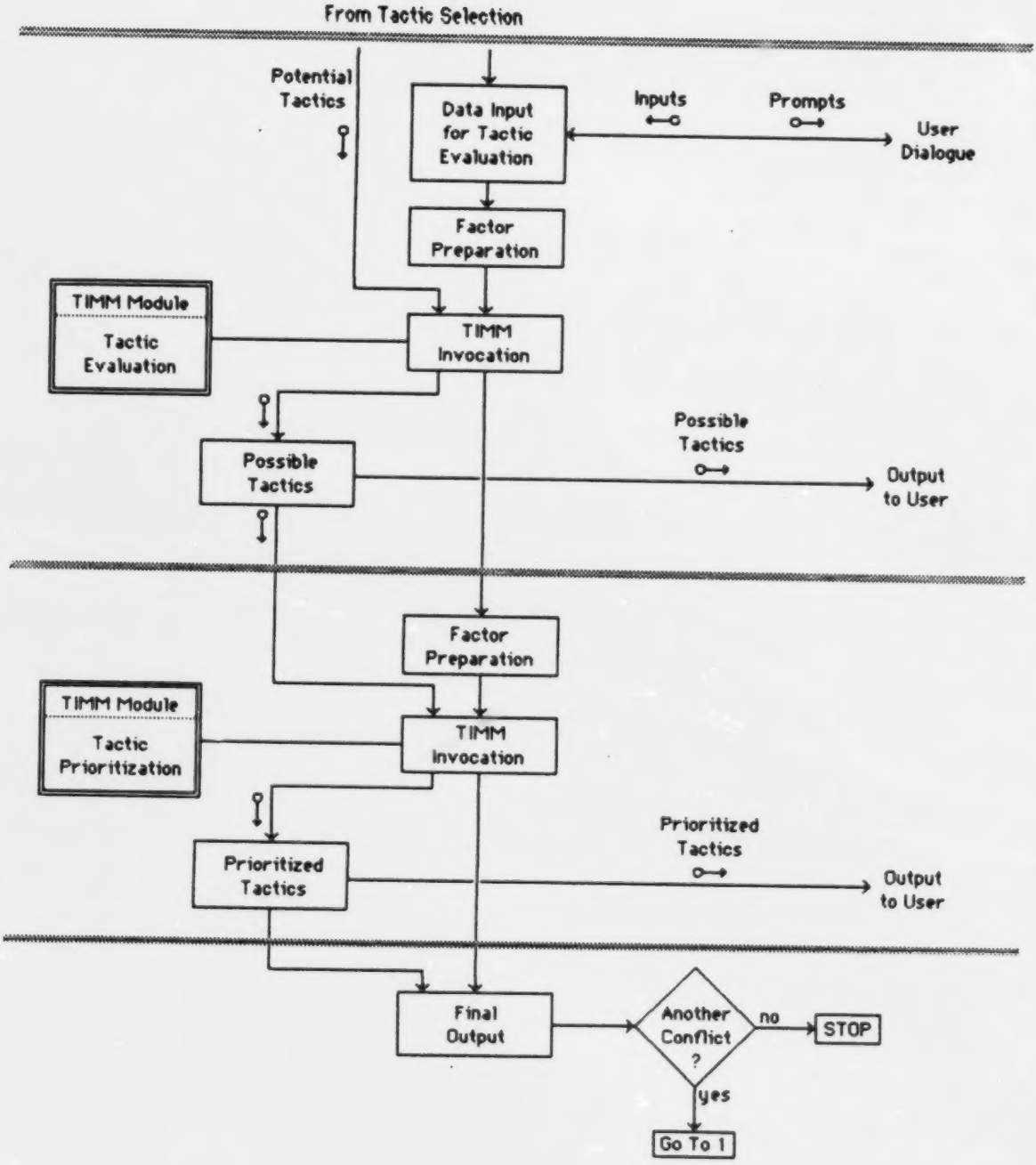
1) Prompt for data input.
2) Show the user the input data, and allow changes.
3) Calculate TIMM factors.
4) Invoke TIMM knowledge modules.
5) Show the results to the user.



CompEngServ Ltd	Revision : 1.0	Figure 9	Shell Structure
	Date : Feb 6, 1986	Drn By : J.D. Beaton	



CompEngServ Ltd	Revision : 1.0	Figure 10	Shell Operation 1
	Date : Feb 6, 1986	Drn By : J.D. Beaton	48



CompEngServ Ltd	Revision : 1.0	Figure 11	Shell Operation 2
	Date : Feb 6, 1986	Drn By : J.D. Beaton	

Note that the system's logical architecture could be made transparent to the user by preparing all of the data before any processing takes place, thus allowing the actual conflict resolution to proceed at the target processor's own speed.

10. VALIDATION TESTS

Validation testing was performed by providing problems for the system to solve that showed it's correct operation and demonstrated particular aspects of it's capabilities. These results were approved by domain experts as accurately representing the conflict avoidance that an air traffic controller would do. In the future, the system may be exercised using problems from the NASD simulator, and compared with actual solutions provided by air traffic controllers.

The test problems solved by the system, and the results that were generated, are shown in Figures 12 through 17. Each conflict type was formally tested twice. The first test was an unconstrained case, where only the aircraft characteristics and the tactic's base weights affected the result. The second test was constrained by various aircraft actions and airspace factors. Each test diagram is divided into three parts. The top section shows the input data that defines the conflict situation. This data is prompted for by the system, and may be entered directly. The middle section shows the conflict avoidance system's results. Since all of the tactics shown are not implemented for all conflicts, the second column contains a check mark for each valid conflict-tactic pair. An "x" indicates that either this conflict-tactic pair is not implemented, or it is not feasible (e.g. "Change Airspeed, Increase" will not resolve a Head-On conflict). The bottom section contains a description of the test, and comments relevant to the test.

Test # 1	Conflict Type Overtaking	
	Aircraft A	Aircraft B
ID	CF-18	B767
Airspeed	610	580
Flight Level	268	270
Attitude	level	level
Next Maneuver	nothing	nothing
Priority	avg.	avg.
Pre-Conflict Angle	0	
Post-Conflict Angle	0	

Track Above	not occ.
Track Below	not occ.
Track Ahead	not occ.
Track to Port of A	not occ.
Track to Starboard of A	not occ.
Weather Conditions	OK
Restricted Airspace	none

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	pass	pass	1.000
2	Change Flight Level Up - B	✓	pass	pass	0.700
3	Change Flight Level Down - A	x	x	x	x
4	Change Flight Level Down - B	x	x	x	x
5	Change Airspeed, Increase - B	✓	pass	pass	0.700
6	Change Track to Port - A	✓	pass	pass	1.000
7	Change Track to Port - B	✓	pass	pass	0.700

Comments:

1) Unconstrained case - result depends only on aircraft characteristics.
2) All priorities are the base priorities for each tactic.

Test # 2	Conflict Type Overtaking			
	Aircraft A	Aircraft B		
ID	CF-18	C5A Galaxy	Track Above	not occ.
Airspeed	800	450	Track Below	occ.
Flight Level	268	270	Track Ahead	occ.
Attitude	level	level	Track to Port of A	occ.
Next Maneuver	nothing	Turn R	Track to Starboard of A	not occ.
Priority	avg.	avg.		
Pre-Conflict Angle	0		Weather Conditions	OK
Post-Conflict Angle	0		Restricted Airspace	none

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	pass	pass	1.000
2	Change Flight Level Up - B	✓	pass	pass	0.504
3	Change Flight Level Down - A	x	x	x	x
4	Change Flight Level Down - B	x	x	x	x
5	Change Airspeed, Increase - B	✓	fail	x	x
6	Change Track to Port - A	✓	pass	fail	x
7	Change Track to Port - B	✓	fail	x	x

Comments:

1) Constrained case.
2) Tactic 5 fails because it is unable to increase it's airspeed enough.
3) Tactic 7 fails tactic selection because aircraft B's next maneuver is a turn to starboard.
4) Tactic 6 fails evaluation because the track to port is occupied.
5) The priority of tactic 2 is affected by it's low maneuverability and the fact that it is above it's optimum altitude.
6) The track below being occupied does not affect this conflict's resolution.

Test # 3	Conflict Type Head-On	

	Aircraft A	Aircraft B
ID	Cessna	C5A Galaxy
Airspeed	170	400
Flight Level	140	150
Attitude	level	level
Next Maneuver	nothing	nothing
Priority	avg.	avg.
Pre-Conflict Angle	180	
Post-Conflict Angle	180	

Track Above	not occ.
Track Below	not occ.
Track Ahead	not occ.
Track to Port of A	not occ.
Track to Starboard of A	not occ.
Weather Conditions	OK
Restricted Airspace	none

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	pass	pass	0.900
2	Change Flight Level Up - B	✓	pass	pass	0.800
3	Change Flight Level Down - A	x	x	x	x
4	Change Flight Level Down - B	x	x	x	x
5	Change Airspeed, Increase - B	x	x	x	x
6	Change Track to Port - A	✓	fail	x	x
7	Change Track to Port - B	✓	pass	pass	0.800

Comments:

1) Unconstrained case - results depend only on aircraft characteristics.
2) Tactic 6 failed selection because it's navigational ability is low, and is unable to navigate this turn.
3) The priority of tactic 1 is reduced because aircraft A (the Cessna) is above it's optimum altitude.
4) The priority of tactic 2 is reduced because the maneuverability of the C5A Galaxy is "low".
5) The priority of tactic 7 is reduced because the maneuverability of the C5A Galaxy is "low".

Test # 4	Conflict Type Head-On			
	Aircraft A	Aircraft B		
ID	Cessna	C5A Galaxy	Track Above	not occ.
Airspeed	170	400	Track Below	not occ.
Flight Level	150	150	Track Ahead	not occ.
Attitude	level	level	Track to Port of A	occ.
Next Maneuver	nothing	nothing	Track to Starboard of A	occ.
Priority	avg.	avg.		
Pre-Conflict Angle	180		Weather Conditions	OK
Post-Conflict Angle	180		Restricted Airspace	to port

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	fail	x	x
2	Change Flight Level Up - B	✓	pass	pass	0.800
3	Change Flight Level Down - A	x	x	x	x
4	Change Flight Level Down - B	x	x	x	x
5	Change Airspeed, Increase - B	x	x	x	x
6	Change Track to Port - A	✓	fail	x	x
7	Change Track to Port - B	✓	pass	fail	x

Comments:

1) Constrained case.
2) Tactic 1 fails selection because the Cessna is at it's maximum altitude.
3) Tactic 6 fails selection because of the Cessna's low navigational capability
4) Tactic 7 fails evaluation because there is restricted airspace to port.
5) The priority of tactic 2 is reduced because the maneuverability of the C5A Galaxy is "low".

Test # 5	Conflict Type Level Crossing Track			
	Aircraft A	Aircraft B		
ID	B767	CF-18	Track Above	not occ.
Airspeed	610	700	Track Below	not occ.
Flight Level	270	270	Track Ahead	not occ.
Attitude	level	level	Track to Port of A	not occ.
Next Maneuver	nothing	nothing	Track to Starboard of A	not occ.
Priority	avg.	avg.		
Pre-Conflict Angle	90		Weather Conditions	OK
Post-Conflict Angle	90		Restricted Airspace	none

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	pass	pass	1.000
2	Change Flight Level Up - B	✓	pass	pass	0.700
3	Change Flight Level Down - A	✓	pass	pass	0.900
4	Change Flight Level Down - B	✓	pass	pass	0.630
5	Change Airspeed, Increase - B	x	x	x	x
6	Change Track to Port - A	x	x	x	x
7	Change Track to Port - B	x	x	x	x

Comments:

1) Unconstrained case - result depends only on aircraft characteristics.
2) The priority of tactic 2 is it's base priority. It is assumed that aircraft B will arrive at the conflict point first, so moving it is less desireable than moving aircraft A due to the first-come-first-served rule.
3) The priority of tactic 3 is reduced because the B767 is below it's optimum altitude, and reducing it's altitude further will make it's operation even more inefficient.
4) The priority of tactic 4 is reduced because the CF-18 is below it's optimum altitude, and it has a base priority of less than one due to the first-come-first-served rule.

Test # 6	Conflict Type	Level Crossing Track		
	Aircraft A	Aircraft B		
ID	B767	C5A Galaxy	Track Above	not occ.
Airspeed	610	450	Track Below	occ.
Flight Level	250	250	Track Ahead	occ.
Attitude	level	level	Track to Port of A	occ.
Next Maneuver	descend	nothing	Track to Starboard of A	occ.
Priority	avg.	avg.		
Pre-Conflict Angle	90		Weather Conditions	OK
Post-Conflict Angle	90		Restricted Airspace	none

Results:

	Tactics		Selection	Evaluation	Prioritization
1	Change Flight Level Up - A	✓	fail	x	x
2	Change Flight Level Up - B	✓	pass	pass	0.504
3	Change Flight Level Down - A	✓	pass	fail	x
4	Change Flight Level Down - B	✓	fail	x	x
5	Change Airspeed, Increase - B	x	x	x	x
6	Change Track to Port - A	x	x	x	x
7	Change Track to Port - B	x	x	x	x

Comments:

1) Constrained case.
2) Tactic 1 fails selection because aircraft A's next maneuver is to descend - don't tell it to climb if it will soon be descending.
3) Tactic 4 fails selection because if it descends, it will still conflict with aircraft A, which will descend soon.
4) Tactic 3 fails evaluation because the track below them is occupied
5) The priority of tactic 2 is reduced because:
   a) It's base priority is 0.7 (first come first served).
   b) It's maneuverability is low.
   c) It is already above it's optimum altitude.

CompEngServ Ltd	Revision : 1.0	Figure 17	Constrained Crossing Track Conflict Test
	Date : Mar 17, 1986	Drn By : J.D. Beaton	

11. EXTENSION TO DENSE AIRSPACES

A dense airspace implies, by our definition, that resolving one conflict could create another conflict. An easy extension of this suggests that the process could continue indefinitely and create a situation where the number of conflicts grows without bound. A formal consideration of the problem suggests that it is unsolvable because it is an instance of the so called n-body problem which, with its exponential increase in computational complexity, has confounded algorithmic approaches in the past.

However, non-algorithmic solutions may be possible. First of all, the problem does not consist of n bodies (i.e. aircraft) in practice, and second, Controllers solve the problem on a regular basis. Therefore, an expert system approach might prove feasible. It was this problem and this approach that we had in mind when the proposal was first submitted.

The approach to expert systems design used by CompEngServ, as was illustrated in this report, is an extension to that normally found in the literature (or textbooks). We do not view a system as a monolithic knowledge base, but as a network of functional activites called a Global Inferencing Algorithm. This approach encourages a wider look at the problem domain and yields a hybrid system which corresponds to the way that an Expert achieves solutions. Such an approach was used to design the system for sparse airspaces. For dense airspaces, the global inferencing algorithm would simulate the practices of a Controller as he resolves conflicts. This involves capturing not only the heuristics but also the meta-knolwedge brought to bear on the solution. This meta-knowledge involves many factors. It is obvious, perhaps, that the Controller applies different sets of rules depending on almost all factors under the general heading of a global state description, and also time. Thus, solutions can be considered as they influence future time states, each of which can be used to find an acceptable solution (for that increment of time). It is proposed that a set (a string through time) of such solutions can be found in any practical situation.

The introduction of time into the expert system involves a host of new definitions and descriptions that have not appeared in the literature before. CompEngServ has developed such a model which might yield a global inferencing algorithm to solve the dense space problem. The opportunity to develop this model should prove of interest to the Ministry.

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

APPENDIX A: KNOWLEDGE MODULE RULES

This appendix contains the rules for each of the TIMM knowledge modules used in this prototype expert system. The rule bases for each of the logical architecture's functions are grouped together, and are preceded by a summary of that function's decisions and factors.

File Name : CLASS

Purpose : Conflict Type

Decision : Conflict Type

Choices :
1) Overtake
2) Head-on
3) Level Same Track
4) Level Crossing Track
5) Level Reciprocal Track
6) Level Merge
7) Ascending Same Track
8) Ascending Reciprocal Track
9) Descending Same Track
10) Descending Reciprocal Track
11) Invalid

Factors

Factor 1 : Relative Flight Level
Values :
1) A > B
2) A = B
3) A < B

Factor 2 : Pre-conflict Relative Heading
Values :
1) Identical
2) Same
3) Crossing
4) Reciprocal
5) Opposite

Factor 3 : Post-conflict Relative Heading
Values :
1) Identical
2) Same
3) Crossing
4) Reciprocal
5) Opposite

Factor 4 : Aircraft A Altitude
Values :
1) Level
2) Descending
3) Ascending

Factor 5 : Aircraft B Altitude
Values :
1) Level
2) Descending
3) Ascending

Factor 6 : Relative Groundspeed
Values :
1) A > B
2) A = B
3) A < B

Decision	Factors					
	Relat Flight Level	Pre-confl Relative Heading	Post-conf Relative Heading	A Atitude	B Atitude	Relative Speed
Overtake	A = B	identical	identical	level	level	A > B
Head-on	A = B	opposite	opposite	level	level	*
lvl same	A = B	same	same	level	level	*
lvl cross	A = B	crossing	crossing	level	level	*
lvl recip	A = B	reciprocl	*	level	level	*
lvl merge	A = B	same	identical	level	level	*
asc same	A > B	same	*	level	ascend	*
asc recip	A > B	reciprocl	*	level	ascend	*
dsc same	A < B	same	*	level	descend	*
asc recip	A < B	reciprocl	*	level	descend	*

Decision	Factors					
	Relat Flight Level	Pre-confl Relative Heading	Post-conf Relative Heading	A Atitude	B Atitude	Relative Speed
invalid	*	*	*	descend	descend	*
invalid	*	*	*	ascend	ascend	*
invalid	*	*	*	ascend	descend	*
invalid	*	*	*	descend	ascend	*
invalid	*	*	*	ascend	level	*
invalid	*	*	*	descend	level	*
invalid	A = B	*	*	level	ascend	*
invalid	A < B	*	*	level	ascend	*
invalid	A = B	*	*	level	descend	*
invalid	A > B	*	*	level	descend	*
invalid	A > B	*	*	level	level	*
invalid	A < B	*	*	level	level	*
invalid	*	identical	opposite	*	*	*
invalid	*	opposite	identical	*	*	*
invalid	A = B	identical	identical	level	level	A < B
invalid	A = B	identical	identical	level	level	A = B

TACTIC SELECTION FACTORS

Factor 1 : A/B Next Maneuver
Values : 1) Ascend
2) Descend
3) Turn L
4) Turn R
5) Nothing

Factor 2 : B/A Next Maneuver
Values : 1) Ascend
2) Descend
3) Turn L
4) Turn R
5) Nothing

Factor 3 : A's or B's Priority
Values : 1) High
2) Avg
3) Low

Factor 4 : Is (A or B) Able to Ascend ?
Values : 1) Yes
2) No

Factor 5 : A's Groundspeed > B's Maximum Groundspeed ?
Values : 1) Yes
2) No

Factor 6 : Is B Able to Increase Airspeed ?
Values : 1) Yes
2) No

Factor 7 : A's or B's Navigation Capabilities
Values : Low
Nominal

Factor 8 : Is A/B Able to Ascend ?
Values : 1) Yes
2) No

File Name : OSCHFLU

Overtake, tactic selection, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : A/B next maneuver (use A when applying to A)
B/A next maneuver (use B when applying to B)
A's/B's priority (use A when applying to B)
Is A/B able to ascend (use A when applying to B)

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	Is A/B Able to Ascend
No	*	*	*	no
No	descend	*	*	*
No	*	ascend	*	*
No	*	*	high	*
Yes	*	*	low	yes
Yes	ascend	descend	avg	yes
Yes	ascend	turn L	avg	yes
Yes	ascend	turn R	avg	yes
Yes	ascend	nothing	avg	yes
Yes	turn L	descend	avg	yes
Yes	turn L	turn L	avg	yes
Yes	turn L	turn R	avg	yes
Yes	turn L	nothing	avg	yes
Yes	turn R	descend	avg	yes
Yes	turn R	turn L	avg	yes
Yes	turn R	turn R	avg	yes
Yes	turn R	nothing	avg	yes
Yes	nothing	descend	avg	yes
Yes	nothing	turn L	avg	yes
Yes	nothing	turn R	avg	yes
Yes	nothing	nothing	avg	yes

File Name : OSCHASIB

Overtake, tactic selection, change airspeed, B only.

Purpose : Use for the conflict type Overtake.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : Is A's Groundspeed > B's Maximum Groundspeed ?
Is B Able to Increase Airspeed ?

Decision	Factors	
	A's airspeed > B's max groungspeed	Is B Able to increase airspeed ?
No	*	no
No	yes	*
Yes	no	yes

File Name : OSCHTRP

Overtake, tactic selection, change trck to port, A or B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : A/B next maneuver
B/A next maneuver
A's/B's priority
A's/B's Navigation Capability

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	A/B Navigational Capability
No	Turn R	*	*	*
No	*	turn L	*	*
No	*	*	high	*
No	*	*	*	low
Yes	*	*	low	nominal
Yes	ascend	ascend	avg	nominal
Yes	ascend	descend	avg	nominal
Yes	ascend	turn R	avg	nominal
Yes	ascend	nothing	avg	nominal
Yes	descend	ascend	avg	nominal
Yes	descend	descend	avg	nominal
Yes	descend	turn R	avg	nominal
Yes	descend	nothing	avg	nominal
Yes	turn L	ascend	avg	nominal
Yes	turn L	descend	avg	nominal
Yes	turn L	turn R	avg	nominal
Yes	turn L	nothing	avg	nominal
Yes	nothing	ascend	avg	nominal
Yes	nothing	descend	avg	nominal
Yes	nothing	turn R	avg	nominal
Yes	nothing	nothing	avg	nominal

File Name : HSCHFLU

Head-on, tactic selection, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Head-on.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : A/B next maneuver (use A when applying to A)
B/A next maneuver (use B when applying to B)
A's/B's priority (use A when applying to B)
Is A/B able to ascend (use A when applying to B)

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	Is A/B Able to Ascend
No	*	*	*	no
No	descend	*	*	*
No	*	ascend	*	*
No	*	*	high	*
Yes	*	*	low	yes
Yes	ascend	descend	avg	yes
Yes	ascend	turn L	avg	yes
Yes	ascend	turn R	avg	yes
Yes	ascend	nothing	avg	yes
Yes	turn L	descend	avg	yes
Yes	turn L	turn L	avg	yes
Yes	turn L	turn R	avg	yes
Yes	turn L	nothing	avg	yes
Yes	turn R	descend	avg	yes
Yes	turn R	turn L	avg	yes
Yes	turn R	turn R	avg	yes
Yes	turn R	nothing	avg	yes
Yes	nothing	descend	avg	yes
Yes	nothing	turn L	avg	yes
Yes	nothing	turn R	avg	yes
Yes	nothing	nothing	avg	yes

File Name : HSCHTRP

Head-on, tactic selection, change trck to port, A or B.

Purpose : Use for both aircraft A and B in the conflict Head-on.

Decision : Should this tactic he applied ?

Choices : Yes
No

Factors : A/B next maneuver
B/A next maneuver
A's/B's priority
A's/B's Navigation Capability

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	A/B Navigational Capability
No	Turn R	*	*	*
No	*	turn R	*	*
No	*	*	high	*
No	*	*	*	low
Yes	*	*	low	nominal
Yes	ascend	ascend	avg	nominal
Yes	ascend	descend	avg	nominal
Yes	ascend	turn L	avg	nominal
Yes	ascend	nothing	avg	nominal
Yes	descend	ascend	avg	nominal
Yes	descend	descend	avg	nominal
Yes	descend	turn L	avg	nominal
Yes	descend	nothing	avg	nominal
Yes	turn L	ascend	avg	nominal
Yes	turn L	descend	avg	nominal
Yes	turn L	turn L	avg	nominal
Yes	turn L	nothing	avg	nominal
Yes	nothing	ascend	avg	nominal
Yes	nothing	descend	avg	nominal
Yes	nothing	turn L	avg	nominal
Yes	nothing	nothing	avg	nominal

File Name : CSCHFLU

Level Crossing Track, tactic selection, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : A/B next maneuver (use A when applying to A)
B/A next maneuver (use B when applying to B)
A's/B's priority (use A when applying to B)
Is A/B able to ascend (use A when applying to B)

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	Is A/B Able to Ascend
No	*	*	*	no
No	descend	*	*	*
No	*	ascend	*	*
No	*	*	high	*
Yes	*	*	low	yes
Yes	ascend	descend	avg	yes
Yes	ascend	turn L	avg	yes
Yes	ascend	turn R	avg	yes
Yes	ascend	nothing	avg	yes
Yes	turn L	descend	avg	yes
Yes	turn L	turn L	avg	yes
Yes	turn L	turn R	avg	yes
Yes	turn L	nothing	avg	yes
Yes	turn R	descend	avg	yes
Yes	turn R	turn L	avg	yes
Yes	turn R	turn R	avg	yes
Yes	turn R	nothing	avg	yes
Yes	nothing	descend	avg	yes
Yes	nothing	turn L	avg	yes
Yes	nothing	turn R	avg	yes
Yes	nothing	nothing	avg	yes

File Name : CSCHFLD

Level Crossing Track, tactic selection, change flight level down, A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Should this tactic be applied ?

Choices : Yes
No

Factors : A/B next maneuver (use A when applying to A)
B/A next maneuver (use B when applying to B)
A's/B's priority (use A when applying to B)
Is A/B able to ascend (use A when applying to B)

Decision	Factors			
	A/B Next Maneuver	B/A Next Maneuver	A/B Priority	Is A/B Able to descend
No	*	*	*	no
No	descend	*	*	*
No	*	ascend	*	*
No	*	*	high	*
Yes	*	*	low	yes
Yes	ascend	descend	avg	yes
Yes	ascend	turn L	avg	yes
Yes	ascend	turn R	avg	yes
Yes	ascend	nothing	avg	yes
Yes	turn L	descend	avg	yes
Yes	turn L	turn L	avg	yes
Yes	turn L	turn R	avg	yes
Yes	turn L	nothing	avg	yes
Yes	turn R	descend	avg	yes
Yes	turn R	turn L	avg	yes
Yes	turn R	turn R	avg	yes
Yes	turn R	nothing	avg	yes
Yes	nothing	descend	avg	yes
Yes	nothing	turn L	avg	yes
Yes	nothing	turn R	avg	yes
Yes	nothing	nothing	avg	yes

TACTIC EVALUATION FACTORS

Factor 1 : Higher Track
Values : 1) Occupied
2) Not occupied

Factor 2 : Track to Port (A or B)
Values : 1) Occupied
2) Not occupied

Factor 3 : Weather Conditions
Values : 1) Bad to port of A
2) Bad to starboard of A
3) Bad above
4) OK
5) Bad below

Factor 4 : Restricted Airspace
Values : 1) To port of A
2) To starboard of A
3) None

Factor 5 : Track ahead
Values : 1) Occupied
2) Not occupied

Factor 6 : Weather Condition to Port of A/B
Values : 1) Bad
2) OK

Factor 7 : Restricted Airspace to port of A/B
Values : 1) Yes
2) No

Factor 8 : Lower Track
Values : 1) Occupied
2) Not occupied

File Name : OECHFLU

Overtake, tactic evaluation, change flight level up, A and B

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Higher track
Weather conditions

Decision	Factors	
	Higher Track	Weather Conditions
No	occupied	*
No	*	bad above
Yes	not occupied	OK
Yes	not occupied	bad to port
Yes	not occupied	bad to starboard
Yes	not occupied	bad below

File Name : OECHASIB

Overtake, tactic evaluation, change airspeed by increasing it, for B only.

Purpose : Use for B only in the conflict Overtake.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Track ahead

Decision	Factors
	Track ahead
No	occupied
Yes	not occupied

File Name : OECHTRP

Overtake, tactic evaluation, change track to port, A and B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Track to port of A/B
Weather conditions
Restricted airspace

Decision	Factors		
	Track to Port of A/B	Weather Conditions	Restricted Airspace
No	occupied	*	*
No	*	bad to port	*
No	*	*	to port
Yes	not occupied	bad to starboard	to starboard
Yes	not occupied	bad to starboard	none
Yes	not occupied	bad above	to starboard
Yes	not occupied	bad above	none
Yes	not occupied	OK	to starboard
Yes	not occupied	OK	none
Yes	not occupied	bad below	to starboard
Yes	not occupied	bad below	none

File Name : HECHFLU

Head-on, tactic evaluation, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Head-on.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Higher track
Weather conditions

Decision	Factors	
	Higher Track	Weather Conditions
No	occupied	*
No	*	bad above
Yes	not occupied	OK
Yes	not occupied	bad to port
Yes	not occupied	bad to starboard
Yes	not occupied	bad below

File Name : HECHTRP

Head-on, tactic evaluation, change track to port, A and B.

Purpose : Use for both aircraft A and B in the conflict Head-on.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Track to port of A/B
Weather conditions to port of A/B
Restricted airspace to port of A/B

Decision	Factors		
	Track to Port of A/B	Weather Conditions to Port of A/B	Restricted Airspace Port of A/B
No	occupied	*	*
No	*	bad	*
No	*	*	yes
Yes	not occupied	OK	no

File Name : CECHFLU

Level Crossing Track, tactic evaluation, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Higher track
Weather conditions

Decision	Factors	
	Higher Track	Weather Conditions
No	occupied	*
No	*	bad above
Yes	not occupied	OK
Yes	not occupied	bad to port
Yes	not occupied	bad to starboard
Yes	not occupied	bad below

File Name : CECHFLD

Level Crossing Track, tactic evaluation, change flight level down, A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Does this tactic apply ?

Choices : Yes
No

Factors : Higher track
Weather conditions

Decision	Factors	
	Lower Track	Weather Conditions
No	occupied	*
No	*	bad below
Yes	not occupied	OK
Yes	not occupied	bad to port
Yes	not occupied	bad to starboard
Yes	not occupied	bad above

TACTIC PRIORITIZATION FACTORS

Factor 1 : Maneuverability
Values : 1) Low
2) Nominal

Factor 2 : Navigation Capability
Values : 1) Low
2) Nominal

Factor 3 : A or B Above Optimum Altitude
Values : 1) Yes
2) No

Factor 4 : Is B at Optimum Airspeed
Values : 1) Yes
2) No

Factor 5 : Next Maneuver to Turn R (A or B)
Values : 1) Yes
2) No

File Name : OPCHFLU

Overtake, tactic prioritization, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
A or B above optimum altitude ?

Factors		Decision and weight	
Maneuverability	A/B Above Optimum Altitude	Yes	No
low	yes	72	28
low	no	80	20
nominal	yes	90	10
nominal	no	100	0

File Name : OPCHASIB

Overtake, tactic prioritization, change airspeed, B only.

Purpose : Use for aircraft B in the conflict Overtake.

Decision : Tactic Priority

Choices : Yes
No

Factors : Is B at optimum airspeed ?

Factors	Decision and weight	
Is B at optimum airspeed	Yes	No
yes	90	10
no	100	0

File Name : OPCHTRP

Overtake, tactic prioritization, change track to port, A or B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
Navigation Capability
Next Maneuver to Turn R (A or B)

Factors			Decision and weight	
Maneuver -ability	Navigation Capability	Next Maneuver to turn R ? (A or B)	Yes	No
low	low	yes	58	42
low	low	no	64	36
low	nominal	yes	72	28
low	nominal	no	80	20
nominal	low	yes	72	28
nominal	low	no	80	20
nominal	nominal	yes	90	10
nominal	nominal	no	100	0

File Name : HPCHFLU

Head-on, tactic prioritization, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Overtake.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
A or B above optimum altitude ?

Factors		Decision and weight	
Maneuverability	A/B Above Optimum Altitude	Yes	No
low	yes	72	28
low	no	80	20
nominal	yes	90	10
nominal	no	100	0

File Name : HPCHTRP

Head-on, tactic prioritization, change track to port, A or B.

Purpose : Use for both aircraft A and B in the conflict Head-on.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
Navigation Capability
Next Maneuver to Turn R (A or B)

Factors			Decision and weight	
Maneuver -ability	Navigation Capability	Next Maneuver to turn R ? (A or B)	Yes	No
low	low	yes	58	42
low	low	no	64	36
low	nominal	yes	72	28
low	nominal	no	80	20
nominal	low	yes	72	28
nominal	low	no	80	20
nominal	nominal	yes	90	10
nominal	nominal	no	100	0

File Name : CPCHFLU

Level Crossing Track, tactic prioritization, change flight level up, A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
A or B above optimum altitude ?

Factors		Decision and weight	
Maneuverability	A/B Above Optimum Altitude	Yes	No
low	yes	72	28
low	no	80	20
nominal	yes	90	10
nominal	no	100	0

File Name : CPCHFLD

Level Crossing Track, tactic prioritization, change flight level down A and B.

Purpose : Use for both aircraft A and B in the conflict Level Crossing Track.

Decision : Tactic Priority

Choices : Yes
No

Factors : Maneuverability
A or B above optimum altitude ?

Factors		Decision and weight	
Maneuverability	A/B Above Optimum Altitude	Yes	No
low	yes	80	20
low	no	72	28
nominal	yes	100	0
nominal	no	90	10

APPENDIX B: FORTRAN SHELL LISTING

This appendix contains a fully commented listing of the FORTRAN shell, which implemented the prototype's user interface and it's control and global data base functions.

```
c*
c*      file name : SHELL.FOR... shell version 2.5
c*

        program shell


        external        REASON
        external        INFO
        external        ACTION

c*************************************************************************
c*
c*      priorities(7) : real array priorities of calculated priorities for
c*                      each of the 7 tactics.
c*      wght : real number for weight of decision made by TIMM.
c*      adjust : real number for weight adjustment factor - set according
c*               tactic being evaluated.
c*
c*      pottac(7) : integer array describing the potential tactics.  Each
c*                  element corresponds to a tactic - an element with value
c*                  of 1 indicates to tactis is potential (else it is 0).
c*      possac(7) : integer array describing the possible tactics.  Each
c*                  element corresponds to a tactic - an element with value
c*                  of 1 indicates to tactis is possible (else it is 0).
c*      lut(7,10) : integer array look up table for potential tactics based
c*                  on conflict type.  Used by conflict evaluation phase to
c*                  set up pottac array.
c*      aircrdata(8,4) : integer array look up table containing aircraft data
c*                       based on aircraft type.
c*      contype : integer number for current conflict type number.
c*      acida, acidb : integer number for aircraft A & B identification
c*                     numbers.
c*      atta, attb : integer number for aircraft A & B altitude values.
c*      mntypa, mntypb : integer number for aircraft A & B next maneuvre
c*                       numbers.
c*      prioa, priob : integer number for aircraft A & B priority numbers.
c*      speeda, speedb : integer number for aircraft A & B ground speed
c*                       values.
c*      alta, altb : integer number for aircraft A & B flight level values.
c*      precangl, postcangl : integer number for post and preconflict angle.
c*      htstat : integer number for higher track status number.
c*      tpstat : integer number for track to port status number.
c*      ltstat : integer number for lower track status number.
c*      weather : integer number for weather condition number.
c*      rastat : integer number for restricted airspace status number.
c*      tastat : integer number for track above status number.
c*      tststat : integer number for track to starboard status number.
c*      maltdif : integer number for minimum altitude difference - used when
c*                deciding if both aircraft are at same flight level.
c*      minalt : integer number containing mimimum altitude an aircraft
c*               can fly while in en-route phase of flight.
c*      factor(15) : integer array used for passing factors to the specific
c*                   TIMM knowledge modules.
c*      temp(15) : integer array used as temporary work array - used when
```

```
c*                calculating factors.
c*       pointer : integer number for factor array index - used in INFO
c*       decision : integer number for decision number made by TIMM.
c*       ISTAT : integer number for the status passed back by TIMM - used
c*              whenever TIMM is envoked.
c*       NFILES : integer number containing the number of TIMM knowledge
c*                module files to be used - always set to 1 !
c*       FIL NAM(10) : character array containing TIMM module file names -
c*                     only first element is used !
c*
c*       tactext(8) : character array containing text for all tactics.
c*       conttxt(11) : character array containing text for all conflict types.
c*       acids(4) : character array containing text for aircraft ID types.
c*       attitud(5) : character array containing text for aircraft attitude
c*                    types.
c*       mantype(5) : character array containing text for aircraft next
c*                    maneuvre types.
c*       priority(3) : character array containing text for aircraft priority
c*                     types.
c*       manuev(2) : character array containing text for aircraft
c*                   maneuvreability types.
c*       htrksts(2) : character array containing text for higher track status
c*                    types.
c*       wthrtyp(5) : character array containing text for weather condition
c*                    types.
c*       rssts(3) : character array containing text for restricted airspace
c*                  types.
c*       outl(20) : character array used for containing all textual data
c*                  to be displayed on the screen.
c*       out2(20) : integer array used for containing all numerical values
c*                  to be displayed on the screen.
c*       response : character variable used for inputting responses from
c*                  keyboard.
c*       digit : boolean value for indicating a numerical input field - used
c*               in PROMPT.
c*       enum : boolean value for indicating a enumeration input field - used
c*              in PROMPT.
c*
c*************************************************************************

      real              priorities(7),wght,adjust

      integer*2         pottac(7),possac(7),lut(7,10)
      integer           aircrdata(8,4),contype,acida,acidb,atta,attb
      integer           mntypa,mntypb,prioa,priob,speeda,speedb,alta
      integer           altb,precangl,postcangl,out2(20),factor(15)
      integer           maltdif,ISTAT,temp(15),NFILES,htstat,tpstat,ltstat
      integer           weather,rastat,tastat,tststat,pointer,decision
      integer           minalt

      character*30      tactext(8),conttxt(11)
      character*20      FIL NAM(10)
      character*12      acids(4),attitud(5),mantype(5)
      character*12      priority(3),manuev(2),htrksts(2),wthrtyp(5)
      character*12      rssts(3),outl(20)
```

```
      character       response

      logical         digit,enum

      common          /b1/ factor,decision,pointer,wght

c*  initialize all variables and arrays

      acida = 1
      acidb = 1
      atta = 1
      attb = 1
      mntypa = 1
      mntypb = 1
      prioa = 1
      priob = 1
      speeda = 0
      speedb = 0
      alta = 0
      altb = 0
      contype = 11
      decision = 1
      htstat = 1
      ltstat = 1
      tpstat = 1
      weather = 1
      rastat = 1
      tastat = 1
      tststat = 1
      precangl = 0
      postcangl = 0
      minalt = 100
      maltdif = 5
      digit = .TRUE.
      enum = .FALSE.

      data pottac /7*0/
      data possac /7*0/
      data priorities /7*0.00/
      data lut /1,1,1,1,1,0,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,
     +1,42*0/

      data NFILES / 1 /
      data FIL NAM / 'CLASS' , 9*' ' /

      data attitud /'level        ','descend     ','ascend      ',
     +'turn R      ','turn L      '/
      data priority /'high         ','avg         ','low         '/
      data mantype /'ascend       ','descend     ','turn L      ',
     +'turn R      ','nothing     '/
      data acids /'B767         ','C5A Gal     ','Cessna      ',
     +'CF-18       '/
      data manuev /'low          ','nominal     '/
      data htrksts /'occupied     ','not occupied'/
      data wthrtyp /'had to port ','had to stbd ','had above   ',
```

```
     +'bad below   ','OK          '/
        data rssts /'to port     ','to sthd     ','none        '/

        data aircrdata /1,650,750,120,290,350,2,2,2,400,500,125,220,300,
     +1,2,3,155,200,65,100,150,2,1,4,850,2000,150,300,500,2,2/

        data conttxt /'Overtaking                    ',
     +'Head-on                    ','Level Same Track              ',
     +'Level Crossing Track       ','Level Reciprocal Track        ',
     +'Level Merge                ','Ascending Same Track          ',
     +'Ascending Reciprocal Track ','Descending Same Track         ',
     +'Descending Reciprocal Track','                             '/

        data tactext /'Increase Altitude - A','Increase Altitude - B',
     +'Increase Airspeed - B','Change Track to Port - A',
     +'Change Track to Port - B','Decrease Altitude - A',
     +'Decrease Altitude - B','                    '/


c*   get data for conflict classification phase and display entries
        write (unit=*,fmt='(''0'',a)')  '** Conflict resolution session is
      + now beginning **'
        write (unit=*,fmt='(''0'',a)')  ' '

30      FIL NAM(1) = 'CLASS'
        write (unit=*,fmt='('' '',a)')  '*** Conflict Classification Phase
      + ***'
        call prompt('Enter aircraft A identification: ',enum,acida,acids,
     +1,4)
        call prompt('Enter aircraft A ground speed: ',digit,speeda,acids,
     +1,2000)
        call prompt('Enter aircraft A flight level: ',digit,alta,acids,1,
     +50000)
        call prompt('Enter aircraft A attitude: ',enum,atta,attitud,1,3)
        call prompt('Enter aircraft B identification: ',enum,acidb,acids,
     +1,4)
        call prompt('Enter aircraft B ground speed: ',digit,speedb,acids,
     +1,2000)
        call prompt('Enter aircraft B flight level: ',digit,altb,acids,1,
     +50000)
        call prompt('Enter aircraft B attitude: ',enum,attb,attitud,1,3)
        call prompt('Enter angle between paths before conflict: ',digit,
     +precangl,acids,0,180)
        call prompt('Enter angle between paths after conflict: ',digit,
     +postcangl,acids,0,180)
c*   set up data for display
        out1(1) = acids(acida)
        out1(2) = acids(acidb)
        out2(1) = speeda
        out2(2) = speedb
        out2(3) = alta
        out2(4) = altb
        out1(3) = attitud(atta)
        out1(4) = attitud(attb)
        out2(5) = precangl
```

```
      out2(6) = postcangl
c*  display accumulated data on screen
      call prntact(pottac,possac,priorities,conttxt(contype),tactext,0)
      call prndat(out1,out2)
      write (unit=*,fmt='(''0'',a)')  'Do you wish to make any changes t
     +o the data ?'
      read (unit=*,fmt='( a)',iostat=IO STATUS) response
      if ((response .eq. 'y') .OR. (response .eq. 'Y')) goto 30
c*  transform data and stuff into array for reason
c*  reset factor array pointer
      pointer = 1
c*  relative flight level
      if ((alta - altb) .gt. maltdif) then
            factor(1) = 1
      else if (IABS(alta - altb) .lt. maltdif) then
            factor(1) = 2
      else if ((alta - altb) .lt. maltdif) then
            factor(1) = 3
      end if
c*  pre-conflict relative heading
      if (precangl .eq. 0) then
            factor(2) = 1
      else if((precangl .ge. 1) .AND. (precangl .le. 45)) then
            factor(2) = 2
      else if((precangl .ge. 46) .AND. (precangl .le. 135)) then
            factor(2) = 3
      else if((precangl .ge. 136) .AND. (precangl .le. 179)) then
            factor(2) = 4
      else if(precangl .eq. 180) then
            factor(2) = 5
      end if
c*  post-conflict relative heading
      if (postcangl .eq. 0) then
            factor(3) = 1
      else if((postcangl .ge. 1) .AND. (postcangl .le. 45)) then
            factor(3) = 2
      else if((postcangl .ge. 46) .AND. (postcangl .le. 135)) then
            factor(3) = 3
      else if((postcangl .ge. 136) .AND. (postcangl .le. 179)) then
            factor(3) = 4
      else if(postcangl .eq. 180) then
            factor(3) = 5
      end if
c*  aircraft A attitude
      factor(4) = atta
c*  aircraft B attitude
      factor(5) = attb
c*  relative ground speed
      if (speeda .gt. speedb) then
            factor(6) = 1
      else if (speeda .eq. speedb) then
            factor(6) = 2
      else if (speeda .lt. speedb) then
            factor(6) = 3
      end if
```

```
c*  make call to reason
        call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
        if (istat .eq. 0) goto 35
        write(unit=*, fmt= '(''0'',a)') 'An error has occured... data must
     + be re-entered.'
        goto 30
c*  set contype to the timm selected value
35      contype = decision
c*  check if valid conflict type was selected
        if (contype .eq. 11) then
                write(unit=*, fmt= '(''0'',a)') 'There is no valid conflic
     +t type... data must be re-entered.'
                goto 30
        end if
c*  conflict type is valid... set potential array from tactics look up table
        do 36 i = 1,7,1
                pottac(i) = lut(i,contype)
36      continue

c*  display accumulated data on screen
        call prntact(pottac,possac,priorities,conttxt(contype),tactext,0)
        call prndat(out1,out2)
c*  wait for user to continue
        write (unit=*,fmt='('' '',a)')  '- Press CR to continue -'
        read (unit=*,fmt='( a)',iostat=IO STATUS) response


c*  get data for tactics selection phase and display the entries
45      write (unit=*,fmt='('' '',a)')  '*** Tactic Selection Phase ***'
        call prompt('Enter aircraft A next maneuver: ',enum,mntypa,
     +mantype,1,5)
        call prompt('Enter aircraft A priority: ',enum,prioa,priority,
     +1,3)
        call prompt('Enter aircraft B next maneuver: ',enum,mntyph,
     +mantype,1,5)
        call prompt('Enter aircraft B priority: ',enum,prioh,priority,
     +1,3)
c*  set up data for display
        out1(5) = mantype(mntypa)
        out1(6) = mantype(mntyph)
        out1(7) = priority(prioa)
        out1(8) = priority(prioh)
c*  display accumulated data on screen
        call prntact(pottac,possac,priorities,conttxt(contype),tactext,0)
        call prndat(out1,out2)
        write (unit=*,fmt='(''0'',a)')  'Do you wish to make any changes t
     +o the data ?'
        read (unit=*,fmt='( a)',iostat=IO STATUS) response
        if ((response .eq. 'y') .OR. (response .eq. 'Y')) goto 45
c*  transform data and stuff into array for reason
c*  reset factor array pointer
        pointer = 1
c*  aircraft A priority
        temp(1) = prioa
c*  aircraft A able to ascend
```

```
        if (alta .lt. aircrdata(6,acida)) then
                temp(2) = 1
        else
                temp(2) = 2
        end if
c*  aircraft B priority
        temp(3) = priob
c*  aircraft B able to ascend
        if (altb .lt. aircrdata(6,acidb)) then
                temp(4) = 1
        else
                temp(4) = 2
        end if
c*  aircraft A able to increase ground speed
        if (speeda .lt. aircrdata(3,acida)) then
                temp(5) = 1
        else
                temp(5) = 2
        end if
c*  aircraft B able to increase ground speed
        if (speedb .lt. aircrdata(3,acidb)) then
                temp(6) = 1
        else
                temp(6) = 2
        end if
c*  aircraft A ground speed > aircraft B max ground speed
        if (speeda .gt. aircrdata(3,acidb)) then
                temp(7) = 1
        else
                temp(7) = 2
        end if
c*  aircraft A navigational capabilities
        temp(8) = aircrdata(8,acida)
c*  aircraft B navigational capabilities
        temp(9) = aircrdata(8,acidb)
c*  aircraft A able to ascend
        if (alta .gt. minalt) then
                temp(10) = 1
        else
                temp(10) = 2
        end if
c*  aircraft B able to ascend
        if (altb .gt. minalt) then
                temp(11) = 1
        else
                temp(11) = 2
        end if
c*  reset potential tactics array pointer and set up factors array
        i = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OSCHFLU'
        else if (contype .eq. 2) then
                FIL NAM(1) = 'HSCHFLU'
        else if (contype .eq. 4) then
                FIL NAM(1) = 'CSCHFLU'
```

```
        else
                stop
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
                factor(1) = mntypa
                factor(2) = mntyph
                factor(3) = temp(1)
                factor(4) = temp(2)
c*  make CHFLU for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*  no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  set up factors array
                factor(1) = mntyph
                factor(2) = mntypa
                factor(3) = temp(3)
                factor(4) = temp(4)
c*  make CHFLU for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*  no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OSCHASIB'
        else
                goto 100
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  set up factors array
                factor(1) = temp(7)
                factor(2) = temp(6)
c*  make SCHASIB for aircraft B call to reason
```

```
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*   no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
100     i = i + 1
c*   reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OSCHTRP'
        else if (contype .eq. 2) then
                FIL NAM(1) = 'HSCHTRP'
        else
                goto 110
        end if
c*   check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*   set up factors array
                factor(1) = mntypa
                factor(2) = mntyph
                factor(3) = temp(1)
                factor(4) = temp(8)
c*   make SCHTRP for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*   no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
110     i = i + 1
c*   reset factor array pointer
        pointer = 1
c*   check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*   set up factors array
                factor(1) = mntyph
                factor(2) = mntypa
                factor(3) = temp(3)
                factor(4) = temp(9)
c*   make SCHTRP for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*   no error... set potential array based on timm decision, update pointer
```

```
                else
                        pottac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 4) then
                FIL NAM(1) = 'CSCHFLD'
        else
                goto 111
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  set up factors array
                factor(1) = mntypa
                factor(2) = mntyph
                factor(3) = temp(1)
                factor(4) = temp(10)
c*  make SCHFLD for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*  no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  set up factors array
                factor(1) = mntyph
                factor(2) = mntypa
                factor(3) = temp(3)
                factor(4) = temp(11)
c*  make SCHFLD for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 45
c*  no error... set potential array based on timm decision, update pointer
                else
                        pottac(i) = decision
                end if
        end if
c*  display accumulated data on screen
111     call prntact(pottac,possac,priorities,conttxt(contype),tactext,1)
        call prndat(out1,out2)
c*  wait for user to continue
```

```
      write (unit=*,fmt='('' '',a)')  '- Press CR to continue -'
      read (unit=*,fmt='( a)',iostat=IO STATUS) response

50    write (unit=*,fmt='('' '',a)')  '*** Tactic Evaluation Phase ***'
      call prompt('Enter track above status: ',enum,htstat,htrksts,
     +1,2)
      call prompt('Enter track below status: ',enum,ltstat,htrksts,
     +1,2)
      call prompt('Enter track ahead status: ',enum,tastat,htrksts,
     +1,2)
      call prompt('Enter track to port of A status: ',enum,tpstat,
     +htrksts,1,2)
      call prompt('Enter track to starboard of A status: ',enum,tststat,
     +htrksts,1,2)
      call prompt('Enter weather conditions: ',enum,weather,wthrtyp,
     +1,5)
      call prompt('Enter restricted airspace status: ',enum,rastat,
     +rssts,1,3)

c*  set up data for display
      outl(10) = htrksts(htstat)
      outl(11) = htrksts(tpstat)
      outl(12) = wthrtyp(weather)
      outl(13) = rssts(rastat)
      outl(14) = htrksts(tastat)
      outl(15) = htrksts(tststat)
      outl(16) = htrksts(ltstat)
c*  display accumulated data on screen
      call prntact(pottac,possac,priorities,conttxt(contype),tactext,1)
      call prndat(outl,out2)
      write (unit=*,fmt='(''0'',a)')  'Do you wish to make any changes t
     +o the data ?'
      read (unit=*,fmt='( a)',iostat=IO STATUS) response
      if ((response .eq. 'y') .OR. (response .eq. 'Y')) goto 50
c*  calculate factors
c*  restricted airspace to port A
      if (rastat .eq. 1) then
            temp(1) = 1
      else
            temp(1) = 2
      end if
c*  restricted airspace to port B
      if (rastat .eq. 2) then
            temp(2) = 1
      else
            temp(2) = 2
      end if
c*  weather conditions to port A
      if (weather .eq. 1) then
            temp(3) = 1
      else
            temp(3) = 2
      end if
c*  weather conditions to port B
      if (weather .eq. 2) then
```

```
                temp(4) = 1
        else
                temp(4) = 2
        end if
c*  reset factor array pointer
        pointer = 1
c*  reset potential tactics array pointer and set up factors array
        i = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OECHFLU'
        else if (contype .eq. 2) then
                FIL NAM(1) = 'HECHFLU'
        else if (contype .eq. 4) then
                FIL NAM(1) = 'CECHFLU'
        else
                stop
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
                factor(1) = htstat
                factor(2) = weather
c*  make ECHFLU for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
                factor(1) = htstat
                factor(2) = weather
c*  make ECHFLU for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OECHASIB'
```

```
                factor(1) = tastat
        else
                goto 101
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  make ECHASIB for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
101     i = i + 1
c*  reset factor array pointer
        pointer = 1
        factor(1) = tpstat
        if (contype .eq. 1) then
                FIL NAM(1) = 'OECHTRP'
                factor(2) = weather
                factor(3) = rastat
        else if (contype .eq. 2) then
                FIL NAM(1) = 'HECHTRP'
                factor(2) = temp(3)
                factor(3) = temp(1)
        else
                goto 120
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  make ECHTRP for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
120     i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                factor(1) = tpstat
                factor(2) = weather
                factor(3) = rastat
        else if (contype .eq. 2) then
                factor(1) = tststat
                factor(2) = temp(4)
```

```
                factor(3) = temp(2)
        else
                goto 121
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  make ECHTRP for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
121     i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 4) then
                FIL NAM(1) = 'CECHFLD'
        else
                goto 122
        end if
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  make ECHFLD for aircraft A call to reason
                factor(1) = ltstat
                factor(2) = weather
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
                        possac(i) = decision
                end if
        end if
122     i = i + 1
c*  reset factor array pointer
        pointer = 1
c*  check if tactic is a potential tactic
        if (pottac(i) .eq. 1) then
c*  make ECHFLD for aircraft B call to reason
                factor(1) = ltstat
                factor(2) = weather
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured... d
     +ata must be re-entered.'
                        goto 50
c*  no error... set possible array based on timm decision, update pointer
                else
```

```
                        possac(i) = decision
                    end if
            end if
c*  display accumulated data on screen
            call prntact(pottac,possac,priorities,conttxt(contype),tactext,1)
            call prndat(out1,out2)
c*  wait for user to continue
            write (unit=*,fmt='('' '',a)')  '- Press CR to continue -'
            read (unit=*,fmt='( a)',iostat=IO STATUS) response


c*  do tactic prioritization
            write (unit=*,fmt='(''0'',a)') '*** Tactic Prioritization Phase **
           +*'
            write (unit=*,fmt='(''0'',a)')  ' '
c*  reset potential tactics array pointer and set up factors array
            i = 1
            pointer = 1
            if (contype .eq. 1) then
                    FIL NAM(1) = 'OPCHFLU'
                    adjust = 1.0
            else if (contype .eq. 2) then
                    FIL NAM(1) = 'HPCHFLU'
                    adjust = 1.0
            else if (contype .eq. 4) then
                    FIL NAM(1) = 'CPCHFLU'
                    adjust = 1.0
            else
                    stop
            end if
c*  check if tactic is a possible tactic
            if (possac(i) .eq. 1) then
                    factor(1) = aircrdata(7,acida)
                    if (alta .gt. aircrdata(5,acida)) then
                            factor(2) = 1
                    else
                            factor(2) = 2
                    end if
c*  make PCHFLU for aircraft A call to reason
                    call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                    if (ISTAT .ne. 0) then
                    write(unit=*, fmt= '(''0'',a)') 'An error has occured in P
           +CHFLU for aircraft A... aborting execution !'
                            stop
c*  no error... set priorities array based on timm decision, update pointer
                    else
                            priorities(i) = wght * adjust
                    end if
            end if
            i = i + 1
c*  reset factor array pointer
            pointer = 1
            if (contype .eq. 1) then
                    adjust = 0.7
            else if (contype .eq. 2) then
```

```
                adjust = 1.Ø
        else if (contype .eq. 4) then
                adjust = Ø.7
        else
                stop
        end if
c*  check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*  set up factors array
                factor(1) = aircrdata(7,acidb)
                if (alth .gt. aircrdata(5,acidb)) then
                        factor(2) = 1
                else
                        factor(2) = 2
                end if
c*  make PCHFLU for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,Ø,INFO,ACTION,ISTAT)
                if (ISTAT .ne. Ø) then
                write(unit=*, fmt= '(''Ø'',a)') 'An error has occured in P
     +CHFLU for aircraft B... aborting execution !'
                        stop
c*  no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
        i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OPCHASIB'
                adjust = Ø.7
        else
                goto 1Ø2
        end if
c*  check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*  set up factors array
                if (speedb .eq. aircrdata(2,acidb)) then
                        factor(1) = 1
                else
                        factor(1) = 2
                end if
c*  make PCHASIB for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,Ø,INFO,ACTION,ISTAT)
                if (ISTAT .ne. Ø) then
                write(unit=*, fmt= '(''Ø'',a)') 'An error has occured in P
     +CHASIB for aircraft B... aborting execution !'
                        stop
c*  no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
1Ø2     i = i + 1
```

```
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                FIL NAM(1) = 'OPCHTRP'
                adjust = 1.0
        else if (contype .eq. 2) then
                FIL NAM(1) = 'HPCHTRP'
                adjust = 1.0
        else
                goto 103
        end if
c*  check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*  set up factors array
                factor(1) = aircrdata(7,acida)
                factor(2) = aircrdata(8,acida)
                if (mntypa .eq. 4) then
                        factor(3) = 1
                else
                        factor(3) = 2
                end if
c*  make PCHTRP for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured in P
     +CHTRP for aircraft A... aborting execution !'
                        stop
c*  no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
103     i = i + 1
c*  reset factor array pointer
        pointer = 1
        if (contype .eq. 1) then
                adjust = 0.7
        else if (contype .eq. 2) then
                adjust = 1.0
        else
                goto 104
        end if
c*  check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*  set up factors array
                factor(1) = aircrdata(7,acidb)
                factor(2) = aircrdata(8,acidb)
                if (mntypb .eq. 4) then
                        factor(3) = 1
                else
                        factor(3) = 2
                end if
c*  make PCHTRP for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
```

```
                write(unit=*, fmt= '(''0'',a)') 'An error has occured in P
     +CHTRP for aircraft B... aborting execution !'
                        stop
c*   no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
104     i = i + 1
c*   reset factor array pointer
        pointer = 1
        if (contype .eq. 4) then
                FIL NAM(1) = 'CPCHFLD'
        else
                goto 105
        end if
c*   check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*   set up factors array
                adjust = 1.0
                factor(1) = aircrdata(7,acida)
                if (alta .gt. aircrdata(5,acida)) then
                        factor(2) = 1
                else
                        factor(2) = 2
                end if
c*   make PCHFLD for aircraft A call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured in P
     +CHTRP for aircraft A... aborting execution !'
                        stop
c*   no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
105     i = i + 1
c*   reset factor array pointer
        pointer = 1
c*   check if tactic is a possible tactic
        if (possac(i) .eq. 1) then
c*   set up factors array
                adjust = 0.7
                factor(1) = aircrdata(7,acidb)
                if (altb .gt. aircrdata(5,acidb)) then
                        factor(2) = 1
                else
                        factor(2) = 2
                end if
c*   make PCHFLD for aircraft B call to reason
                call REASON(1,NFILES,FIL NAM,0,INFO,ACTION,ISTAT)
                if (ISTAT .ne. 0) then
                write(unit=*, fmt= '(''0'',a)') 'An error has occured in P
     +CHTRP for aircraft A... aborting execution !'
```

```
                        stop
c*  no error... set priorities array based on timm decision, update pointer
                else
                        priorities(i) = wght * adjust
                end if
        end if
c*  display accumulated data on screen
        call prntact(pottac,possac,priorities,conttxt(contype),tactext,1)
        call prndat(out1,out2)
c*  wait for user to continue
        write (unit=*,fmt='(''0'',a)')  'Do you wish to resolve another co
     +nflict ?'
        read (unit=*,fmt='( a)',iostat=IO STATUS) response
        if ((response .eq. 'y') .OR. (response .eq. 'Y')) then
                do 98 i = 1,20,1
                        out1(i) = '                '
98              continue
                do 99 i = 1,7,1
                        pottac(i) = 0
                        possac(i) = 0
                        priorities(i) = 0.00
99              continue
                contype = 11
                goto 30
        end if
        write (unit=*,fmt='('' '',a)')  '** Conflict resolution session is
     + now ending **'
        stop

        end
```

```
c*************************************************************************
c*                                                                       *
c*   PROMPT(ptxt, type, val, choices, min, max)                          *
c*      ptxt = prompt text string                                        *
c*      type = value type: numerical = .TRUE., enum. = .FALSE.           *
c*      val = returned integer entered by user                           *
c*      choices = array of choices text strings                          *
c*      min,max = min and max boundary values for entered integer        *
c*                                                                       *
c* This subroutine prints the question. If the value is enumeration      *
c* type it will print the choices in order on the screen and the         *
c* user will have to give the corresponding choice number as an          *
c* answer. However, if the value is a number then the choices            *
c* window is cleared and the user will have to enter numerical           *
c* value.                                                                *
c*                                                                       *
c* CALLS : NONE                                                          *
c*                                                                       *
c* CALLED BY : MAIN                                                      *
c*                                                                       *
c*************************************************************************
```

```
        subroutine prompt(ptxt,type,val,choices,min,max)

        character*(*)    ptxt,choices(*)
        logical          type
        integer*2        row
        integer          val,min,max,IO STATUS,index,i

4       if (type) then
c*  type = numerical... print choices array
        write (unit=*, fmt= '('' '',a17,i4,a1)') '(current value = ',val
     +,')'
        else
c*  type = enumeration... print choices array
        write (unit=*, fmt= '('' '',a)') 'Choices are:'
        do 2 i = min,max,1
                if (i .eq. val) then
                        marker = 16
                else
                        marker = 32
                end if
                write (unit=*,fmt= '('' '',1x,a1,t5,a1,a3,a)') marker,
     +          i+48,' -  ',choices(i)
2       continue
        end if
c*  display prompt text and wait for entry
1       write (unit=*, fmt= '('' '',a)') ptxt
        read (unit=*, fmt='( i4)', iostat=IO STATUS) index
        if (index .lt. 0) then
                return
        else
                val = index
        end if
        if ((val .ge. min) .AND. (val .le. max)) then
                return
c*  entered value is out of bounds... error
        else
        write (unit=*, fmt= '(''0'',a)') 'invalid response... try again'
                goto 4
        end if

        end



c*************************************************************************
c*                                                                       *
c*      PRNDAT()                                                         *
c*                                                                       *
c* This routine prints a summary of the entered data on the screen       *
c*                                                                       *
c* CALLED BY : MAIN                                                      *
c*                                                                       *
c*************************************************************************
```

```
      subroutine prndat(11,12)

      integer          12(*)
      character*12     11(*)


      write(unit=*,fmt=6) 221,'   AIRCRAFT A    ',221,'    AIRCRAFT B
     + ',221,'   CONFLICT   ',221,'        GENERAL',222
      write( unit=*,fmt=20 ) 221,'-----------------------------------
     +-----------------------------------',222
      write(unit=*,fmt=7) 221,'ID : ',11(1),221,'ID : ',11(2),221,
     +'PRCA: ',12(5),221,'HTS: ',11(10),222
      write(unit=*,fmt=8) 221,'GS : ',12(1),221,'GS : ',12(2),221,
     +'POCA: ',12(6),221,'TBS: ',11(16),222
      write(unit=*,fmt=9) 221,'FL : ',12(3),221,'FL : ',12(4),221,221,
     +'TAS: ',11(14),222
      write(unit=*,fmt=10) 221,'ATT: ',11(3),221,'ATT: ',11(4),221,221,
     +'TPS: ',11(11),222
      write(unit=*,fmt=10) 221,'NM : ',11(5),221,'NM : ',11(6),221,221,
     +'TSS: ',11(15),222
      write(unit=*,fmt=10) 221,'PRI: ',11(7),221,'PRI: ',11(8),221,221,
     +'WEA: ',11(12),222
      write(unit=*,fmt=11) 221,221,221,221,'RAS: ',11(13),222
      write( unit=*,fmt=20 ) 221,'-----------------------------------
     +-----------------------------------',222
      return

6     format(' ',a1,a,a1,a,a1,a,a1,a,t80,a1)
7     format(' ',a1,2x,2a,t20,a1,2x,2a,t39,a1,2x,a,i3,t54,a1,2x,2a,t80,
     +  a1)
8     format(' ',a1,2x,a,i4,t20,a1,2x,a,i4,t39,a1,2x,a,i3,t54,a1,2x,2a,
     +  t80,a1)
9     format(' ',a1,2x,a,i4,t20,a1,2x,a,i4,t39,a1,t54,a1,2x,2a,t80,a1)
10    format(' ',a1,2x,2a,t20,a1,2x,2a,t39,a1,t54,a1,2x,2a,t80,a1)
11    format(' ',a1,t20,a1,t39,a1,t54,a1,2x,2a,t80,a1)
20    format(' ',a1,a,a1)
      end


c*************************************************************************
c*                                                                       *
c*      PRNTACT()                                                        *
c*                                                                       *
c* This routine prints the tactics data on the screen                    *
c*                                                                       *
c* CALLED BY : MAIN                                                      *
c*                                                                       *
c*************************************************************************

      subroutine prntact(potac,posac,prio,t1,t2,flag)

      character*30     t2(*),t1
      integer*2        potac(*),posac(*),i,flag
      real             prio(*)
```

```
      write( unit=*,fmt=19 ) 221,'-----------------------------------
     +-----------------------------------',222
      write(unit=*,fmt='('' '',a1,a,a30,t80,a1)' ) 221,' CONFLICT TYPE:
     +',t1,222
      write(unit=*,fmt='('' '',a1,a,t80,a1)') 221,' TACTICS:',222
      if (flag .eq. 1) then
            do 1 i = 1,7,1
            if ((potac(i) .eq. 1) .AND. (posac(i) .eq. 1)) then
                  write(unit=*,fmt=4) 221,16,t2(i),prio(i),222
            else if ((potac(i) .eq. 1) .AND. (posac(i) .ne. 1)) then
                  write(unit=*,fmt=5) 221,32,t2(i),222
            end if
1           continue

      write(unit=*,fmt='('' '',a1,t80,a1)') 221,222
      write(unit=*,fmt='('' '',a1,5x,a,a1,a,t80,a1)') 221,'[ ',16,' on L
     +HS is possible tactic; number on RHS is priority level. ]',222
      else
            write(unit=*,fmt='('' '',a1,t80,a1)') 221,222
      end if
      write( unit=*,fmt=19 ) 221,'-----------------------------------
     +-----------------------------------',222
      return

4     format(' ',a1,t15,a1,1x,a30,1x,f5.3,t80,a1)
5     format(' ',a1,t15,a1,1x,a30,t80,a1)
19    format(' ',a1,a,a1)

      end


c*************************************************************************
c*                                                                       *
c*   INFO(FAC NAM,MIN VAL,MAX VAL,ITYPE,FAC VAL,RESRVD,NVALS,IVAL,       *
c*        WEIGHT,ISTAT)                                                  *
c*      fac nam = factor name string... do not change !!                 *
c*      min val = minimun factor value... do not change !!               *
c*      max val = maximum factor value... do not change !!               *
c*      itype = factor type... do not change !!                          *
c*      fac val = possible factor values array... do not change !!       *
c*      resrvd = reserved                                                *
c*      nvals = number of selected values... always set to 1             *
c*      ival = selected values array                                     *
c*      weight = weight of selected values... not used                   *
c*      istat = output status... always set to 12                        *
c*                                                                       *
c* CALLS : NONE                                                          *
c*                                                                       *
c* CALLED BY : REASON                                                    *
c*                                                                       *
c*************************************************************************

      subroutine INFO(FAC NAM,MIN VAL,MAX VAL,ITYPE,FAC VAL,RESRVD,
```

```
     +NVALS,IVAL,WEIGHT,ISTAT)

         character*(*)    FAC NAM
         integer          MIN VAL,MAX VAL,ITYPE,ISTAT,NVALS,IVAL(*),val
         integer          factor(15),pointer,decision
         character*(*)    FAC VAL(*)
         real             RESRVD,WEIGHT(*),wght

         common           /b1/ factor,decision,pointer,wght


         val = factor(pointer)
         if ((val .lt. MIN VAL) .OR. (val .gt. MAX VAL)) then
c*  value not within timm boundaries... print error and abort
                  write(unit=*,fmt='(''0'',a)') 'Error in INFO... factor out
     + of bounds - aborting'
                  ISTAT = 16
                  return
         end if
c*  all okay... set values and return
         NVALS = 1
         ISTAT = 12
         IVAL(1) = val
         pointer = pointer + 1
         return

         end


c*************************************************************************
c*                                                                       *
c*       ACTION( FAC NAM,MIN VAL,MAX VAL,ITYPE,FAC VAL,RELY,NVALS,IVAL,  *
c*               WEIGHT,ISTAT )                                          *
c*                                                                       *
c*       fac nam = factor name string... do not change !!                *
c*       min val = minimun factor value... do not change !!              *
c*       max val = maximum factor value... do not change !!              *
c*       itype = factor type... do not change !!                         *
c*       fac val = possible factor values array... do not change !!      *
c*       resrvd = reserved                                               *
c*       rely = reliability (0 - 1) only when istat = 12                 *
c*       nvals = number of selected decision values                      *
c*       ival = selected decision values array                           *
c*      weight = weight of selected decision values                      *
c*       istat = output status                                           *
c*                                                                       *
c* CALLS : NONE                                                          *
c*                                                                       *
c* CALLED BY : REASON                                                    *
c*                                                                       *
c*************************************************************************


      subroutine ACTION(FAC NAM,MIN VAL,MAX VAL,ITYPE,FAC VAL,RELY,
     +NVALS,IVAL,WEIGHT,ISTAT)
```

```
      character*(*)    FAC NAM,FAC VAL(*)
      integer          MIN VAL,MAX VAL,ITYPE,NVALS,IVAL(*),ISTAT
      integer          decision,pointer,factor(15)
      real             RELY,WEIGHT(*),wght
      character        ANSWER

      common           /b1/ factor,decision,pointer,wght


      if (ISTAT .eq. 10) then
              write (unit=*,fmt='(''0'',a)') 'Action... there is no possi
     +ble choice - aborting'
              ISTAT = 16
              return
      end if

c*    get decision and weight values passed by timm
      decision = IVAL(1)
      if (IVAL(1) .eq. 1) then
              wght = WEIGHT(1)
      else
              wght = WEIGHT(2)
      end if
c*    check reliability value
      if (RELY .ne. 1.00) then
              write (unit=*,fmt='(''0'',5a)') 'The decision ',FAC NAM,
     +'is',FAC VAL(IVAL(1))
              write (unit=*,fmt='('' '',a)') 'is not reliable - continue
     +(c) or redo (r) ?'
              read (unit=*,fmt='( a)',iostat=IO STATUS) ANSWER
              if ((ANSWER .eq. 'c') .or. (ANSWER .eq. 'C')) then
                      ISTAT = 8
              else if ((ANSWER .eq. 'r') .or. (ANSWER .eq. 'R')) then
                      ISTAT = 16
              else
                      ISTAT = 16
              end if
      else
              ISTAT = 8
      end if
      return

      end
```

APPENDIX C IMPLEMENTATION CONSIDERATIONS

The implementation of this system has been affected by a number of factors which will be discussed here.

One of the largest constraints has been the system builder, TIMM, which is an inductive reasoning tool. Its rules are limited to 20 different antecedent factors, each of which is similar to a relational statement for a high-level language variable (i.e. is a stated relation between a variable and some value true or not?). The consequent consists of a single "variable", or decision, which may take on up to fifteen values, each of which may have a certainty factor associated with it. A set of rules with it's antecedents and consequent makes up a module. Modules may be linked, with the consequent of one module either providing an antecedent for another module or being used to call up another module. If more antecedents or consequents are required, then a number of modules must be used. An in-depth type search is used, where a module is exited as soon as any rule fires. The inferencing engine within TIMM always matches what it considers to be the rule that is "closest" to the input data, Therefore, rules are needed to pick up on impossible situations so that a correct answer is always returned to the shell.

TIMM does provide good rule editing facilities, and it does provide a way to add a high level language shell, unlike most system builders priced within this projects budget constraints.

TIMM does not support meta-rules, and requires a considerable high level language programming effort to invoke external procedures. It's rule structure imposes certain types of partitions on the knowledge base, and also limits the granularity of modules. Also, it's search strategy uses a pattern-matching algorithm that always fires the one rule considered "closest" to the set of antecedent factors. Since it is supported by a high level language shell acting as a user interface, it is sufficient for this prototype system. However, a larger system would require a more flexible builder.

Time constraints have dictated that this system implement a subset of all of the conflict types and resolution tactics available. These have been chosen to illustrate the presented concepts as best as possible, and they provide a subset that gives sufficient information and examples so that completing the rule bases would be a fairly straight forward task.

Note that processing may well be a critical factor in future systems. The structure of this prototype addresses this well, because it offers a large potential for parallelism at different

system levels. Each of the five functions in Figure 3 could be pipelined, so that multiple conflicts could be resolved much faster. Each rule in the Conflict Classification rule base may be traversed independently, and therefore in parallel. The Tactic Selection rule base's tactic partitions and conflict type sub-partions provide an enormous number of modules that could be searched in parallel. The Tactic Evaluation and Prioritization functions could be applied to each potential tactic independently, with the different factors each function uses providing additional processing partitions. These partitions will result in a very large degree of parallelism within each system function, which will provide the flexibility necessary to devise an efficient multiple-processor system architecture for a large scale conflict avoidance system.

hold this data should be created whenever a tactic is selected. Many of this frame's slots could be filled in based only on conflict data, such as required maneuvers and their execution times. This will be the start of an unambiguous tactic specification, to be completed during Tactic Evaluation (the next function). It will also provide this next function with any specific resolution data it may need. Note that procedural attachments will be required to perform the necessary calculations.

4.2.4 Tactic Evaluation

Implemented

When given a set of tactics that could potentially solve the conflict, the Tactic Evaluation function must eliminate those tactics that can not be used to solve the conflict in the present airspace configuration. The result is a set of possible tactics, i.e. those tactics for which it is possible to determine a complete conflict resolution procedure.

The potential tactics (the input) are consistent and feasible only with respect to the conflict itself. The evaluation of these conflict-specific tactics involves considering all other factors that might affect the implementation of this tactic (i.e. constraints). This would include the location of other aircraft, possible interferences caused by planned maneuvers, airspace restrictions, maximum cruising altitudes, etc. In order to determine some of this data, especially detecting situations like aircraft that will prevent certain maneuvers, classical software routines would be invoked by antecedents within the rule base.

Proposed

The possible tactics resulting from Tactic Evaluation could be an empty set if the airspace imposes enough constraints. However, there must always be a solution for any conflict situation. Therefore, it would be beneficial to consider constraints as being either hard or soft. Hard constraints are those that can not possibly be relaxed, while soft constraints are those that are desirable rather than required. If the Tactic Evaluation function outputs an empty set, some number of the soft constraints could be relaxed to some degree, and the inputs could be reprocessed. This relaxation could continue until a non-empty set is output.

This function would complete the modification of each tactic's frame, again based only on non-conflict data. As an input, the tactic frame would provide the specific information necessary to evaluate the tactic's feasibility with respect to both other aircraft and the airspace in general.